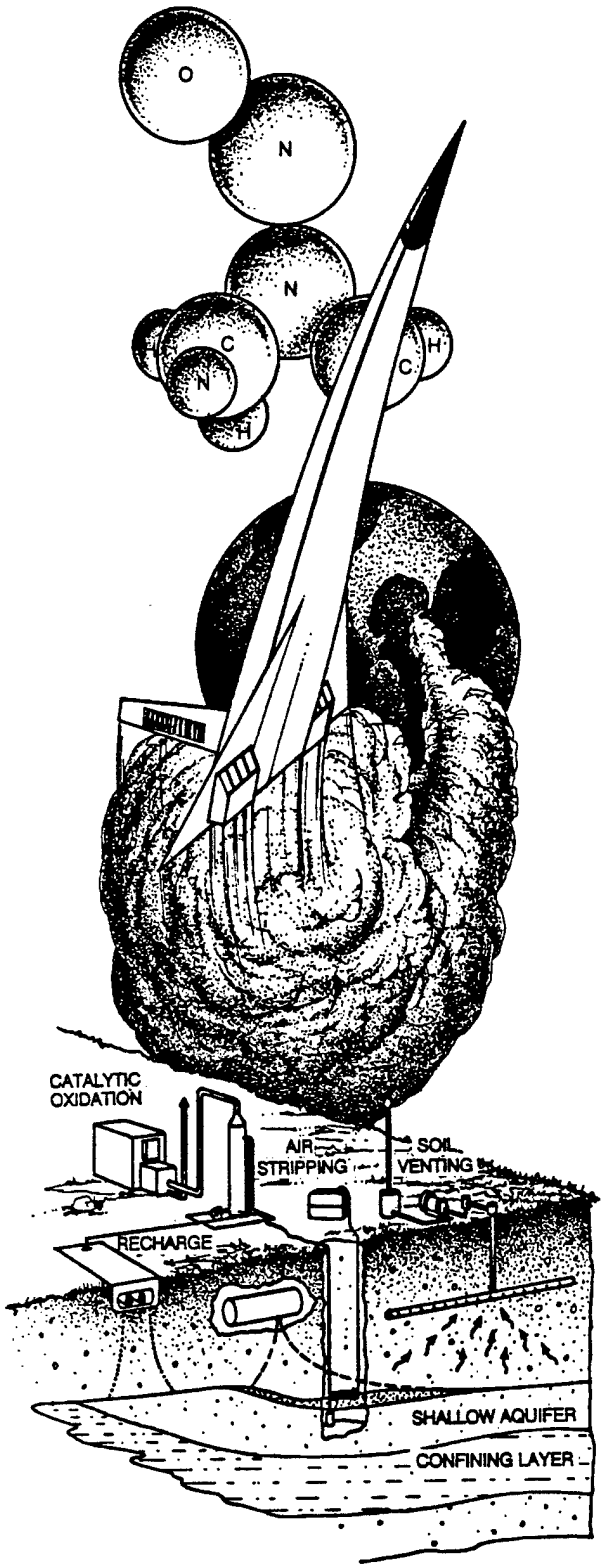
ESL-TR-92-13



APPLICATION OF PATTERN RECOGNITION TECHNIQUES TO PROBLEMS IN ADVANCED POLLUTION MONITORING

B.K. LAVINE, A.B. STINE, X.H. QIN

DEPARTMENT OF CHEMISTRY AND
DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
CLARKSON UNIVERSITY
POTSDAM, NY 13699

MAY 1995

FINAL REPORT

NOVEMBER 1989 - DECEMBER 1991

ENVIRONICS DIVISION
Air Force Engineering & Services Center
ENGINEERING & SERVICES LABORATORY
Tyndall Air Force Base, Florida 32403

NOTICE

PLEASE DO NOT REQUEST COPIES OF THIS REPORT FROM HQ AFESC/RD (ENGINEERING AND SERVICES LABORATORY), ADDITIONAL COPIES MAY BE PURCHASED FROM:

NATIONAL TECHNICAL INFORMATION SERVICE
5285 PORT ROYAL ROAD
SPRINGFIELD, VIRGINIA 22161

FEDERAL GOVERNMENT AGENCIES AND THEIR CONTRACTORS REGISTERED WITH DEFENSE TECHNICAL INFORMATION CENTER SHOULD DIRECT REQUESTS FOR COPIES OF THIS REPORT TO:

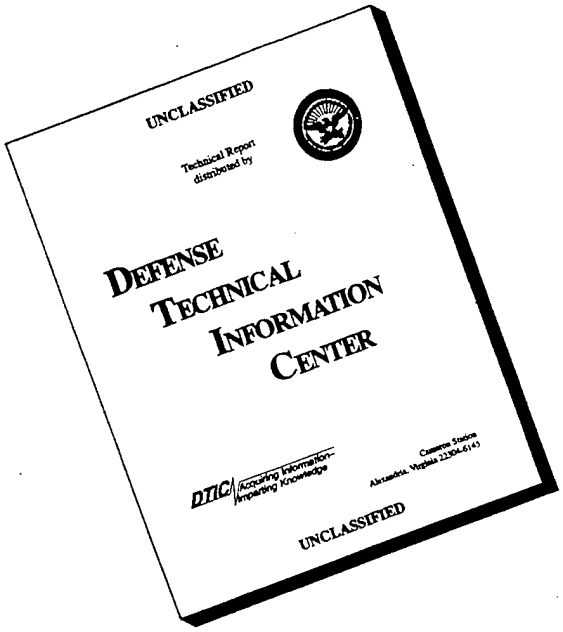
DEFENSE TECHNICAL INFORMATION CENTER
CAMERON STATION
ALEXANDRIA, VIRGINIA 22314

The following commercial products (requiring Trademark (tm)) are mentioned in this report. Because of the frequency of usage, the Trademark was not indicated. If it becomes necessary to reproduce a segment of this document containing any of these names, this notice must be included as part of that reproduction.

Unix	VAXstation	SunOS	OSF/Motif
VAX	SPARCstation	Solaris	Digital
VMS	MS-Windows 3.0	X Window	DECstation
Sun	MS-DOS	X	Ultrix
XUI	Zenith	DEC	DECwindows
IMSL	Intel	HP	IBM
SGI	OPENLOOK	SPARC	Lotus 1-2-3
			Quattro Pro

Mention of the products listed above does not constitute Air Force and/or Clarkson University endorsement or rejection of this product, and use of information contained herein for advertising purposes without obtaining clearance according to contractual agreements is prohibited.

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final Report, November 1989 - April 1992

4. TITLE AND SUBTITLE

Application of Pattern Recognition Techniques to Problems in Advanced Pollution Monitoring

5. FUNDING NUMBERS

F08635-90-C-105

6. AUTHOR(S)

B. K. Lavine, A. B. Stine, and X. H. Qin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Department of Chemistry &
Department of Electrical and Computer Engineering
Clarkson University
Potsdam, NY 13699

8. PERFORMING ORGANIZATION

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQ AFESC/RDVC
Tyndall AFB, FL 32403

10. SPONSORING/MONITORING

ESL-TR-92-13

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This technical report details the development and implimentation of a pattern recognition technique, termed the Fuzzy-c Varieties (FCV) technique. This technique is intended to classify patterns which exhibit membership in only a single class, or data patterns which may partially represent multiple classes. The ability for the patterns to represent multiple classes permits the technique to be of potential value for classifying environmental analysis patterns of mixed samples, such as samples of mixed fuels. The technique was developed and applied to data patterns representing classification measurements on iris flowers, the Fisher iris data set. It was tested further with data patterns representing gas chromatograms of pure and mixed samples of jet fuels. The FCV classification algorithm was implimented as a computer program, written in the Fortran computer programming language, and using data display capabilities provided by the X-Windows standard graphical user interface. The technical report describes the classification results obtained by the FCV system from the Fisher iris data set and from the jet fuel data sets. The technical report also provides a user's guide to the FCV computer software.

14. SUBJECT TERMS
Pattern Recognition, Mixture Analysis, Gas Chromatography, Classification

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

(The reverse of this page is blank.)

EXECUTIVE SUMMARY

A. OBJECTIVE

To identify the jet fuel and/or fuels responsible for ground water pollution, it is necessary to collect gas chromatographic (GC) data representative of the different types of jet fuels. Since the resulting data base is both large and highly complex, a new methodology for analyzing multivariate data must be developed to ensure accurate pollutant typing. Hence, there is a need for computer programs that can analyze complex multivariate GC profile data sets. These programs must be flexible and robust enough for researchers but sufficiently user friendly for use by so called non-expert technicians. An anticipated application of these computer programs would be the identification of the type of jet fuel present in a recovered environmental sample. The computer program would be queried as to the type of fuel responsible for the chromatographic profile of the sample. Therefore, the computer program in this scenario must possess two key attributes: (1) it must be simple to operate, and (2) the program should be easily portable to a wide variety of hardware platforms which is what has been accomplished in this project.

B. BACKGROUND

The technique of gas chromatography/mass spectrometry (GC/MS) has been used by the United States Air Force to identify environmental pollutants, e.g., jet fuels. Typically, identifications are made on the basis of fingerprint patterns in gas chromatographic data. The characteristic nature of GC/MS data suggests application of pattern recognition techniques to render data interpretation into a more quantitative form, and to allow for more accurate predictions of unknowns.

AFCESA has initiated an in-house research project to examine samples of various jet fuels in order to develop a suitable data base of gas chromatograms representative of the different types of jet fuels. Using conventional pattern recognition techniques, AFESC scientists have shown that fuel samples can be identified as to type on the basis of gas chromatographic data. However, gas chromatograms of jet fuel samples can be a combination of two or more different fuels. Conventional pattern recognition techniques cannot properly treat this type of data. Hence, the development of new pattern recognition methods that can cope with this type of data has been a major thrust of the project. The fuzzy c-varieties (FCV) clustering algorithm and FCV-false color data imaging are our response to AFCESA's unique data analysis problems.

C. SCOPE

Due to the existence of good software tools for program development, initial prototyping was carried out using a DEC VAXstation. Several basic modules were completed as a proof of concept using GKS, in order to validate the design and the overall system concept. The X Window System was chosen as our basis for graphics since GKS was found not to be universally available across a wide variety of platforms. Further development was done using the DEC VAXstation and the X Window System.

During the development timeframe, the Open Software Foundation (a consortium including DEC, IBM, HP, and other major industry) introduced Motif, which was to become a standard graphical user interface (layered on top of the X Window System). We converted the software to use Motif, in order to facilitate porting the system to other hardware platforms, including the Sun SPARCstation. We also ported the system to the Intel 80x86/MS-DOS using Microsoft Windows 3.0, in order to make the system available on that platform, in addition to the workstation platforms.

D. METHODOLOGY

There are basic differences between the FCV clustering algorithm and conventional pattern recognition methods. With conventional methods, every sample in the data set is assigned to only one class, whereas the FCV technique permits each sample in the data set to have a partial membership in each of the different classes (i.e., different fuel types) in the data. Hence, it is possible to determine the contribution of a potential source (i.e., fuel type) to a fuel spill directly from the class membership values. When compared to conventional pattern recognition techniques, the FCV clustering algorithm possesses other advantages including: (1) the capability to analyze data sets with a low object to descriptor ratio, and (2) the capability to tune out noise in the data by judicious selection of algorithm parameters. Clearly, these features of the FCV clustering algorithm can result in a very definite advantage if one is interested in a careful analysis of the data.

The ability of the FCV clustering algorithm to extract information from a data base can be extended by using this algorithm in conjunction with principal component analysis and false color data imaging, i.e., FCV-false color data imaging. FCV-false color data imaging can assess the structural characteristics of a data set by organizing the data into subgroups, clusters or hierarchies. By examining a 3-dimensional map of a data set generated in an FCV-false color data imaging experiment, an investigator can determine via graphics the degree of separation between the different classes in the data and also identify influential outliers present in the data. Hence, FCV false color data imaging is ideally suited for analyzing data bases of the type developed by AFCESA.

E. TEST DESCRIPTION

Three data sets were generated by Dr. Howard Mayfield of AFCESA for testing the proposed software. The first data set (neat fuel data set) consisted of 201 total ion chromatograms representing five different types of jet fuels. The fuel samples were obtained from either Wright Patterson Air Force Base or Mukilteo WA Energy Management Laboratory and constituted a representative sampling of the fuels. The second data set (mixed fuel data set) consisted of 95 total ion chromatograms of five types of jet fuels, as well binary mixtures of various fuels types (i.e., 50/50 mixtures). The total ion chromatograms in this data set were peak matched using the same 76-feature retention time template as was used for the first neat fuel data set. The third data set (water soluble data set) consisted of total ion chromatograms of 94 water samples that contained dissolved hydrocarbons. The object was to determine whether or not total ion chromatograms of these dissolved hydrocarbons could be used to identify the particular type of jet fuel responsible for the contamination. The computer program that was used for peak matching yielded a set of 48 standardized retention time windows.

F. RESULTS

The first study showed that sample variability due to the different manufacturing processes used to generate a particular variety of jet fuel does not interfere with our attempts to identify neat jet fuels. On the other hand, instrumental drift due to changes in the operating conditions of the GC/MS can be a serious problem which must be taken into account when designing a pattern recognition system for fuel typing. This study has also demonstrated the utility of FCV-false color data imaging for uncovering hidden relationships within complex multivariate data sets.

In the second study, the feasibility of identifying composite fuel samples (i.e. the binary mixtures) was demonstrated using the FCV clustering algorithm. The superiority of the FCV clustering algorithm over other principal component based pattern recognition methods, e.g., SIMCA, for the prediction of the class assignments of unknowns was also established in this study.

In the third study gas chromatograms of dissolved hydrocarbons were used to identify the original fuel responsible for the contamination of the water samples. These results show that it is possible to monitor a suspected well via GC/MS and determine the type of fuel responsible for the contamination through pattern recognition analysis of the dissolved hydrocarbon fuel profile.

G. CONCLUSIONS

The results of these three studies show that it is both feasible and profitable to develop a pattern recognition system for source identification of pollutants. Using pattern recognition techniques, an investigator can quantitate the effects (if any) of instrumental variability on classification (typing of fuels) and can also identify recovered environmental samples that contain pollutants from multiple sources. Because the Federal government assigns penalties and assesses economic liability for clean-up and restorative costs to polluters, the capability developed in these studies is both timely and critical.

H. RECOMMENDATIONS

Pattern recognition studies are performed using four different operations: transduction, preprocessing, feature selection, and classification. The work described in this report has been directed towards the development of better classification methods for GC data. However, the confounding of the desired class information (fuel type) by experimental artifacts present in the data is a serious problem that needs to be addressed. Hence, research must be directed towards the development of feature selection methods that minimize these unwanted relationships within the data. Since statistics cannot offer us a solution to this problem save experimental design, brute force calculations based on a nearest neighbor methodology are justified. Such a calculation would identify a set of features that maximize differences between the classes while ensuring that fuel typing is made on the basis of legitimate chemical differences between the different types of fuels. The availability of supercomputing power via workstations (e.g., IBM RISC 6000 Workstations which are available at Clarkson University) makes this approach both attractive and practical.

PREFACE

This report was prepared by Clarkson University, Department of Chemistry, Potsdam NY 13699-5810, under Contract Number F08635-90-C-0105, for the Headquarters Air Force Engineering and Services Center, Directorate of Engineering and Services Laboratory, Division of Environmental Chemistry (HQ AFESC/RDVC), Tyndall Air Force Base, Florida 32403-6001. This work was sponsored by the US Air Force Engineering and Services Center (AFESC). Dr. Howard Mayfield (AFESC/RDVC) was the Government technical manager. This report summarizes work accomplished between 1 November 1989 and 31 December 1991.

This report has been reviewed by the Public Affairs Office and is releasable to the National Technical Information Services (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Howard T. Mayfield, PhD
Project Officer

Michael G. Katona, PhD
Chief Scientist, Environics Directorate

Jimmy C. Cornette, PhD
Chief, Environmental Research Division

Niel J. Lamb, Colonel, USAF, BSC
Director, Environics Directorate

(The reverse of this page is blank.)

TABLE OF CONTENTS

TABLE OF CONTENTS
(CONTINUED)

TABLE OF CONTENTS
(CONCLUDED)

NOTE: TABLES OF CONTENTS ARE LOCATED IN THE FRONT OF INDIVIDUAL APPENDICES

LIST OF FIGURES

LIST OF FIGURES
(CONCLUDED)

NOTE: LISTS OF FIGURES ARE LOCATED IN THE FRONT OF INDIVIDUAL ATTACHMENTS.

LIST OF TABLES

NOTE: LISTS OF TABLES ARE LOCATED IN THE FRONT OF INDIVIDUAL ATTACHMENTS.

LIST OF ABBREVIATIONS

API	Application Programming Interface
C	C programming language
CPU	Central Processing Unit
DIF	Data Interchange Format
FCV	Fuzzy c-Varieties
FORTRAN	Formula Translator
GC	Gas Chromatography
GC/MS	Gas Chromatography/Mass Spectrometry
GL	Graphics Library (Silicon Graphics Inc.)
GKS	Graphical Kernel System
GUI	Graphical User Interface
HPLC	High Performance Liquid Chromatography
MOTIF	Open Software Foundation Graphical User Interface Definition
MSFCV	Microsoft Windows based False Color Data Imaging/Visualization System
OPENLOOK	AT&T/Sun Microsystems Graphical User Interface Definition
OS	Operating system
PC	Principal Component
PCPLOT	Principal Component Plotting
PEXlib	PHIGS Extention for X Window library

LIST OF ABBREVIATIONS (CONCLUDED)

PHIGS	Programmers Hierarchical Interactive Graphics System
RISC	Reduced Instruction Set Computer
SC	Spreadsheet Calculator
SIMCA	Soft Independent Modelling by Class Analogy
VMS	Operating system for DEC VAX computer systems
VGA	Video Graphics Adapter
X	X Window System
XFCV	X Window based False Color Data Imaging/Visualization System
XGRABSC	Public domain X Window Screen Grabber

LIST OF SYMBOLS

D	Distance
L	Discriminant function value
I	Fraction of total cumulative variance
N	Dimensionality of the data matrix
S	Variance-covariance matrix
w	Weight vector
x	Data vector
λ	Eigenvalue
μ	Class membership value
c	Number of clusters to be detected in the data
v	Cluster center
n	Total number of sample in the data set
d	Unit eigenvector
r	Total number of eigenvectors

Subscript

$_k$	Sample number *k*
$_i$	Cluster *i*
$_{ik}$	Cluster center *i* of sample *k*
$_{jk}$	Cluster center *j* of sample *k*
$_{ij}$	*jth* eigenvalue of the within-cluster scatter matrix *i*

LIST OF SYMBOLS
(CONCLUDED)

Superscript

^{T}	Transpose of the data matrix
^{m}	Fixed weighting exponent

GLOSSARY OF TERMINOLOGY

API - Application Programming Interface. This is the definition by which a software system is programmed (i.e., it defines the subroutines and data structures used)

Autoscaling - A linear transformation of the data resulting in each measurement variable having a mean of zero and standard deviation of unity.

DIF - Data Interchage Format. This is the standard data format used by various commercial spreadsheet applications (e.g., Lotus 1-2-3, Quattro-Pro, etc.)

False Color Data Imaging - A technique used by satellites to image data.

Fuzzy c-Variety - An algorithm which employs the concept of partial class membership in the clustering of multivariate data.

Linear Discriminants - A method of classification that employs a linear hyperplane to separate samples represented as data points in a high-dimensional measurement space.

Normalization - A preprocessing method for removing information about sample size from the data.

Pattern Recognition - A set of methods for investigating data represented as points in a high-dimensional measurement space

Picard Iteration - An iterative method which gives an approximate solution to a boundary value problem.

Principal Component Analysis - A method for transforming correlated measurement variables into new, uncorrelated variables which are called principal components.

SIMCA - A method of classification based on eigenvector projections of the data.

Weight Vector - The vector which defines the linear discriminant function.

(The reverse of this page is blank.)

SECTION I

INTRODUCTION

A. OBJECTIVE

This report outlines the development of a new methodology for handling large complex multivariate data sets. Analytical techniques such as gas chromatography/mass spectrometry (GC/MS) or high performance liquid chromatography (HPLC) can be used to characterize organic residues present in complex environmental samples. The data that is generated is high dimensional and often will require pattern recognition techniques for interpretation. However, applications of pattern recognition techniques to problems in chemical analysis are often complicated by three factors: (1) the individual classes in the data are not homogenous, (2) the desired group information is confounded by experimental artifacts, and (3) there are serious colinearities and multicolinearities among the measurement variables used to characterize the data. Several projects involving these effects and methods for dealing with them are discussed. This report also addresses a frequently occurring problem in pattern recognition, namely the analysis of data with a low object to descriptor ratio.

B. BACKGROUND

In the last decade a major effort has been made to substantially improve the analytical methodology applied to the study of complex environmental samples. For example, instrumental methods such as GC/MS and HPLC have greatly increased the number of compounds that can be identified and quantified even at the trace level. This capability has, in turn, allowed scientists to attack ever more complex problems (e.g., source identification of oil spills), but paradoxically has also led to an information handling problem.

The reason for this problem is that in any monitoring effort it is necessary to analyze a large number of samples in order to assess the wide variation in composition that an environmental system possesses. The combination of the large number of samples that must be analyzed and the number of constituent species that must be measured per sample yields data sets of such size and complexity that it is often not possible to understand the nature of the system under investigation without using multivariate statistical methods of analysis. Unfortunately, there has been little research focusing on the development of techniques to handle data generated in such studies. Over the past few years only a few papers have appeared in the literature on this subject (1-4).

Pattern recognition techniques are well suited for analyzing data generated in profiling studies because of the characteristics of the procedures. No exact functional form is fitted to the data; rather, relationships are sought which provide definitions of similarity among diverse groups of data. In essence, pattern recognition techniques can be thought of as providing relations that uncover common properties. Once such relations are developed they may be used to infer the properties of samples that were not part of the original data set. These techniques are also capable of dealing with high-dimensional data, where more than three measurements are used to represent each object. Furthermore, pattern recognition methods can handle data from multiple sources, where each measurement can be the result of a separate, independent experiment. Finally, techniques are available for selecting important features from a large set of parameters. Thus, studies can be performed on systems where the exact relationships are not fully understood.

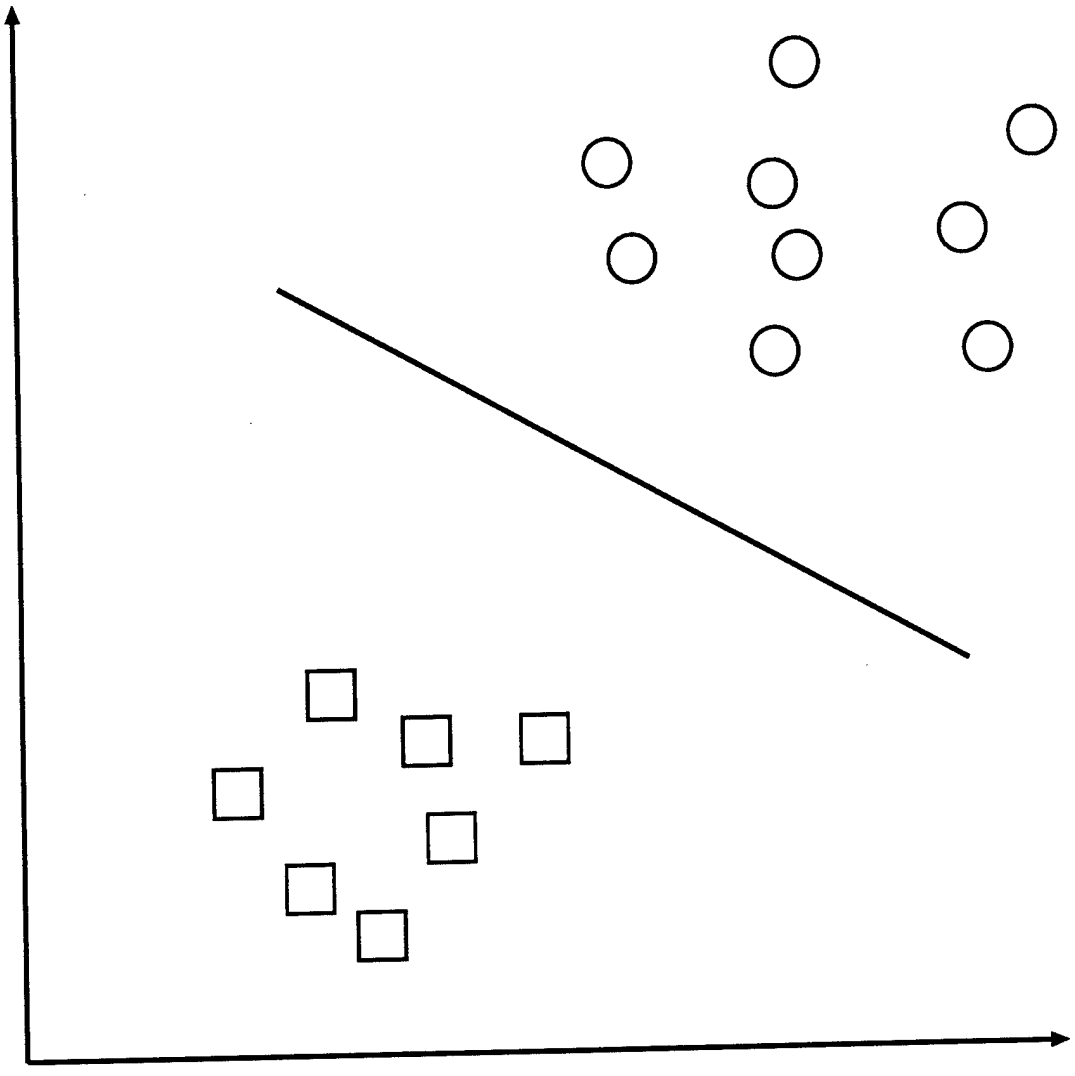


Figure 1. Example of a Linear Discriminant Function Application.

A large body of pattern recognition literature in both science and engineering has focused on *linear discriminant analysis* (5-6). The motivation for using these functions can be seen in Figure 1 for the case of two-dimensional data (two chromatographic parameters in this case). The parameters cause the points to cluster effectively into two groups. A linear discriminant function defines the boundary between the two groups. In the figure this boundary is the line that separates the two groups. For three-dimensional data, the boundary is a plane, while the boundary becomes a hyperplane in the case of multi-dimensional data. The boundary is defined by the vector orthogonal to it. This vector is called the *weight vector*, and the linear discriminant has the form

$$L = x^T w \tag{1}$$

where L is the value of the discriminant function, x is the test vector formed from the chromatographic parameters, and w is the weight vector defining the boundary between the two clusters. The sign of L is used to determine on which side of the boundary the vector lies. The test vector is classified on the basis of this computation.

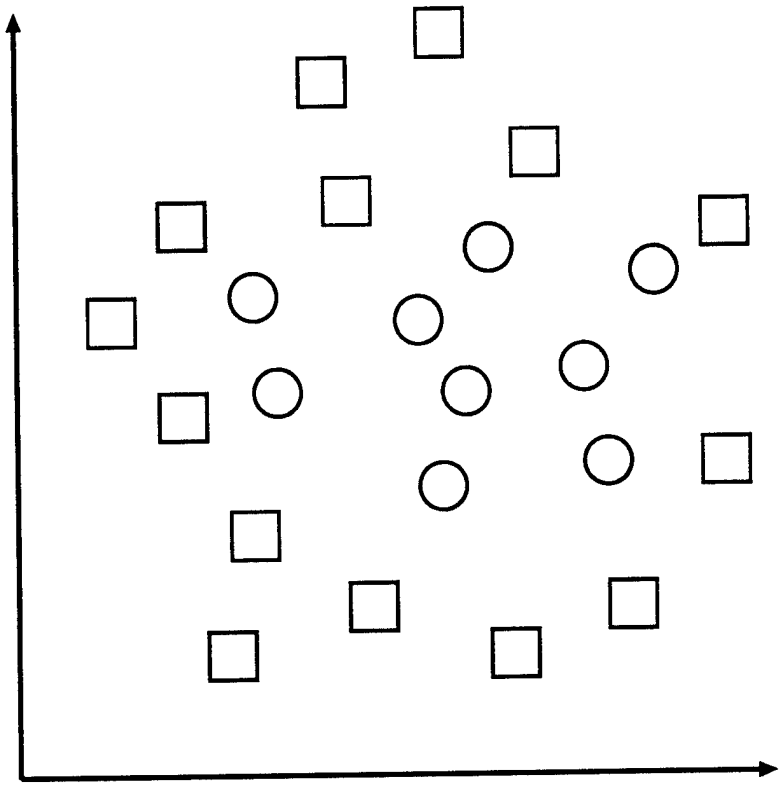


Figure 2. Hypothetical Data Space that is Not Amenable to Description by a Linear Discriminant.

There are three reasons why linear discriminant analysis is not the method of choice for the present application. First, the discriminant divides the data space into two regions. All vectors lie on either one side of the boundary or the other, and class assignments are made on the basis of the boundary. But consider the data space shown in Figure 2. Let the circles be the target class. It is not unreasonable to assume that the target class will cluster in one region of the space, and the other vectors will be distributed randomly throughout the data space. This type of data structure, the so-called *asymmetric case* (7), is sometimes encountered in profiling studies. In this particular case, a linear discriminant would not yield reliable results. This is especially true if a limited amount of data is available for defining the boundary.

The second major problem associated with linear discriminants is chance classification. This can be a serious problem, especially in situations involving nonseparable training sets. Using *Monte Carlo simulation techniques*, Lavine (8,9) has shown that the degree of separation due to chance is dependent upon several factors, e.g., number of descriptors per sample, number of samples in the data set, class membership distribution of the training set data, and covariance structure of the data. Furthermore, these factors do not act independently of one another. Lavine concluded in his study that the basic prerequisite for the successful application of hyperplane methods is a data set where the number of independent samples is far greater than the number of variables, e.g., an object to descriptor ratio of at least five to one and preferably ten to one. Unfortunately, this situation does not always occur in monitoring studies.

The third major problem associated with linear discriminants is colinearity. One of the assumptions made in discriminant analysis is that the predictor variables are truly independent. If this assumption is invalid, the estimated coefficients of the weight vector will be affected by noise and consequently far removed from the target value. Clearly, the problem of colinearity will adversely affects the performance of a linear classifier. Due to the presence of measurement colinearities in profiling data, one should first orthogonalize the data prior to linear discriminant analysis.

Because of the problems associated with linear discriminants, a modelling approach should be used to define the classification rule. The expectation is that if the set of vectors representing the class of interest is described properly, decisions can be made regarding the similarity of the test vector to the vectors that define the cluster. Although there are a number of approaches to modelling clusters, the *fuzzy c-varieties (FCV) clustering algorithms* (10,11) are well suited for analyzing profile data. These clustering algorithms borrow little from actual fuzzy set theory, although the concept of shared-set membership is basic to the operation of these algorithms. The FCV clustering algorithm describes the data using disjoint principal component models. The method is similar to SIMCA (12). However, the FCV clustering algorithm, unlike SIMCA, can identify a sample that is a convex combination of two

or more prototype vectors, (i.e., the test sample is a linear combination of the measurement variables of two or more groups), and it can also estimate the weights defining the convex combination. When compared to conventional pattern recognition techniques, the FCV clustering algorithms possess several advantages: (1) they are not hindered by the presence of colinearities or multicolinearities among the measurement variables in the data set, (2) they are capable of seeking out clusters of different shapes, and (3) class models can be estimated using this algorithm even for data with a low object to descriptor ratio. Fuzzy clustering methods used in conjunction with the techniques of principal component analysis (13) and false color data imaging (14), i.e., *FCV-false color data imaging*, can provide insight into the underlying relationships present in complex multivariate data sets. With this approach the risk of arbitrarily imposing a nonrepresentative structure on the data is reduced.

C. SCOPE/APPROACH

1. Data Analysis

Implementation of principal component analysis and the FCV clustering algorithm is in the form of an interactive modular system written in FORTRAN-77. Modules for data management and scaling have also been developed. Where possible, existing components have been used (e.g., data management using existing spreadsheet programs with interfaces suitable for use with this system).

2. Data Visualization

Software to implement false color data imaging was developed using C and a windowing system appropriate to the target hardware platforms (X Window for workstations and Microsoft Windows for MS-DOS-based personal computers). Again, where practical, existing components were utilized in the system to minimize delivery time for this system (e.g., public-domain screen print utilities were used rather than rewriting one specifically for this system).

SECTION II

SYSTEM OVERVIEW

A. STATISTICAL METHODS

1. Principal Component Analysis

Graphical methods are often used by physical scientists to study data. If there are only two or three measurements per sample, the data can be displayed as a graph for direct viewing. In otherwords, the data can be displayed as points in a two- or three-dimensional measurement space. The coordinate axes of the space are defined by the measurement variables. By examining the graph, a scientist can search for similarities and dissimilarities among the samples, find natural clusters, and even gain information about the overall structure of the data set. If there are n-measurements per sample (n > 3), a two- or three-dimensional representation of the measurement space is needed in order to visualize the relative position of the data points in n-space. This representation must accurately reflect the high dimensional structure of n-space. One such approach is to use a mapping and display technique called principal component analysis.

Principal component analysis is a method for transforming the original measurement variables into new, uncorrelated variables called principal components. Each principal component (PC) is a linear combination of the original measurement variables. Using this procedure is analogous to finding a set of orthogonal axes which represent the directions of greatest variance in the data. (The variance is defined as the degree to which the data points are spread apart in the n-dimensional measurement space.) If a data set has a large number of interrelated variables, then principal component analysis is a powerful method for analyzing the structure of that data and reducing the dimensionality of the pattern vectors.

The procedure for implementing principal component analysis is as follows. First, the covariance matrix of the data set is computed:

$$S=\frac{1}{N-1}X^{T}X \qquad (2)$$

where X is the data matrix, X^T is the transpose of the data matrix, and N is the dimensionality of the data. Next, an eigenanalysis is performed on the covariance matrix. The eigenvector corresponding to the largest eigenvalue represents the

direction of greatest variance in the data; each successive eigenvector represents the direction of maximum residual variance. The eigenvectors (i.e.,principal components) are then arranged in order of decreasing variance - the first eigenvector is the most informative and the last eigenvector is the least informative. For a two-dimensional display, the two largest eigenvectors or principal components are retained; the values of the two largest principal components are computed for each data point. Finally, the points are projected onto a plane defined by the two largest principal components. The amount of information in the principal component plot relative to the original measurement variables is expressed as

$$I = \frac{\lambda_1 + \lambda_2}{\sum_{K=1}^{N} \lambda_K} \qquad (3)$$

where I is the fraction of the total cumulative variance, λ_K is the eigenvalue of the kth eigenvector, and N is the number of descriptors in the data matrix.

2. FCV Clustering Algorithm

The FCV clustering algorithms were first published by Bezdek in 1981. Since that time, they have been successfully used to analyze satellite and star-mapping data (15,16). For profile analysis the method was first used to assess the role of cuticular hydrocarbons in nestmate recognition for social hymenoptera (17) and proved to be a useful tool. The FCV algorithm can be used to search for trends present in a data set and can cluster the data points according to these trends. The algorithm is most useful when the data structure is not known, as is often the case in studies concerned with source apportionment of environmental pollutants.

An interesting feature of the FCV clustering algorithm is that each data vector in the training set is assumed to contribute to the modelling of each of the classes within the data. The actual algorithm consists of solving simultaneously the following set of equations using a *Picard iteration* (18)

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{D_{ik}}{D_{jk}}\right)^{1/(m-1)}} \qquad (4)$$

$$v_i = \frac{\sum_{k=1}^{n} (\mu_{ik})^m x_k}{\sum_{k=1}^{n} (\mu_{ik})^m} \qquad (5)$$

$$S_i = \sum_{k=1}^{n} (\mu_{ij})^m (x_k - v_i)(x_k - v_i)^T \quad (6)$$

$$D_{ik} = \left(|x_i - v_i|^2 - \sum_{j=1}^{r} \langle x_k - v_{ij}, d_{ij} \rangle^2 \right)^{\frac{1}{2}} \quad (7)$$

The membership value of sample k with respect to cluster i ($i = 1, 2, 3, ...$) is μ_{ik}, and these values are subject to the conditions $0 < \mu_{ik} < 1$ and $\Sigma \mu_{ik} = 1$. D_{ik} is the distance of sample k from cluster center i, v_i is the center of cluster i, x_k is the sample data vector, d_{ij} is a unit eigenvector corresponding to the jth largest eigenvalue of the (fuzzy) within-cluster scatter matrix, S_i, and m is a fixed weighting exponent. To obtain an approximate solution to this set of four equations, the user must supply a starting class membership matrix. The cluster centers, the within-cluster scatter matrix for each cluster, and the distance of each sample from each cluster are computed in rapid succession. New membership values are then computed for the samples in the final step of the first iteration. The algorithm continues by using these new membership values as the starting matrix for a second iteration through the same set of equations. This process is allowed to continue until convergence is achieved. The number of iterations required to achieve convergence depends upon the minimum prespecified change criterion for the class membership values.

The value of m is usually set at 2, but by increasing m, less weight is attached to the importance of the samples with smaller membership values. The higher the value of m, the fuzzier the algorithm becomes in the sense that points whose membership values are uniformly low through the iterative procedure tend to become increasingly ignored in determining the membership functions and the defining linear varieties. It is this feature of the FCV algorithm that is particularly appealing when one suspects that the data may not exist in compact well separated clusters. The ability to "tune out" noise in the data by adjusting m can be of great value in obtaining favorable and meaningful clustering results for this type of data.

When investigating the data with the FCV clustering algorithm, one may choose to search for round clusters in the data by specifying $r = 0$ or find the best fit of the data to linear clusters by specifying $r = 1$, or in general attempt to fit the data to other geometric shapes by setting $r \geq 2$. This feature of the FCV algorithm allows the investigator to compare the fit of the data to a number of geometrically distinct cluster shapes and to select the fit which appears to best represent the actual structure of the data. Clearly, this feature results in a very definite advantage if one is interested in a careful analysis of the data.

3. FCV-False Color Data Imaging

If a scientist is seeking the relationship between a physical property and the concentration of a species in a sample, a simple plot of property vs. concentration usually leads to the correct functional relationship. The modern scientist, however, is becoming increasingly involved with experimental design and complex measurement systems (e.g., GC/MS or HPLC). Therefore, data analysis often means finding relationships, not only between the property of interest and the composition of the sample, but also developing relationships between sets of chemical measurements and the class assignment of the sample. This has forced scientists to turn to methods of multivariate data analysis, in order to have any hope of extracting information from the large data matrices generated in such experiments. Unfortunately, a valuable analytical tool has been lost in the process, precisely when it is needed the most: that is, the ability to simply graph the data and directly examine its quality, distribution, and structure. Consequently, there is a demand for techniques which can reduce the dimensionality of a data set down to one, two, or three dimensions.

One approach to this problem involves the development a novel mapping and display method for the purpose of examining the structure of a multivariate data set, utilizing the techniques of *false color data imaging*, *principal component analysis*, and the *fuzzy c-varieties (FCV) pattern recognition clustering algorithm*. The technique of false-color data imaging has long been used to analyze multispectral data measured by satellite electromagnetic scanner systems. In a false-color data imaging experiment, each data vector is projected onto a coordinate system defined by the three principal axes: x, y, and z. Each of the three principal axes is assigned a primary color, e.g., red, green, or blue. A given data point can then be assigned a color which is a combination of the three primary colors. The intensity of each primary color is inversely scaled according to the distance of the data point from the particular color axis in question. Thus, data projected onto a line equidistant from all three coordinate axes would be assigned a color composed of equal intensities of the three primary colors, i.e., some shade of gray.

The first step in an FCV-false color data imaging experiment is to project the original high-dimensional data onto a suitable three-dimensional subspace defined by the three largest principal components of the data. Since this is easily accomplished with microcomputer graphics even for large data sets, it is a reasonable first step in any data analysis. However, there is a problem associated with principal component maps. The spatial relationships between individual data points, and between groups of data points in the original measurement space are often distorted in the projective process. It is at this point where this method departs substantially from other graphical display techniques. By taking advantage of the membership coefficients generated by the FCV clustering algorithm, much of the spatial information lost during projection is restored through the use of color, a crucially important information dimension. Using the FCV clustering algorithm, the data is

fitted to a user-specified number of linear disjoint principal component models. Each model, which represents a different cluster of data points, is assigned a different color: red, green, blue, etc. A given data point is displayed as a combination of these colors, with the amount of any one color determined by the sample's membership value for that particular class. Interpretation of the resulting color images provide valuable insight into the data structure. For example, two projected data points do not lie close to one another in the high-dimensional space (and hence are not similar) unless they are displayed in nearly the same color. A single cluster of data points in the three-dimensional principal component space, that appears to be made up of two distinct primary colors, will be interpreted as two distinct clusters whose true multidimensional separation has been lost through projection. A solid group of data appearing in a non-primary color suggests the presence of an unsuspected class, and so forth.

B. GRAPHICAL VISUALIZATION COMPONENTS

The following hardware and software components were used in the development of the graphical portion of the system, hereafter known as *XFCV* (*MSFCV* on Intel 80x86/MS-DOS/MS-Windows Personal Computer):

1. Hardware

a. DEC VAX/VMS Workstation

DEC VAXstation 3100 Model 38
8 Plane GPX graphics adapter
24M memory
400M disk
16" Color display (resolution 1024x864)
Approx. CPU speed: 5 mips

b. Intel 80x86/MS-DOS Personal Computer

Zenith Z-248 (80286 @ 8MHz)
ATI VGA Wonder graphics adapter w/ 512K video memory
4M memory
40M disk (Seagate ST251-1)
13" NEC MultiSync 3D color display (resolution 1024x768)
Approx. CPU speed: .7 mips

c. DEC RISC/Ultrix Workstation

DEC DECstation 5000 Model 200
8 Plane CX graphics adapter
32M memory
1.8G disk
16" Color display (resolution 1024x864)

Approx. CPU speed: 27 mips

d. SUN Solaris/SPARC Workstation

SUN SPARCstation-2
8 Plane GX graphics adapter
16M memory
400M disk
16" Color display (resolution 1152x900)
Approx. CPU speed: 27 mips

2. Software

a. DEC VAX/VMS Workstation

VAX/VMS V5.3-1
DECwindows V2
DEC DECwindows OSF/Motif V1.1.1 developers toolkit
VAX C V3.1
VAX FORTRAN V5.4

b. Intel 80x86/MS-DOS Personal Computer

MS-DOS V5.0
Microsoft Windows 3.0
Borland C++ V2.1
Microsoft Fortran V5.1

c. DEC RISC/Ultrix Workstation

Ultrix 4.x (tested with 4.0, 4.1, 4.2)
DEC C 1.0
DEC FORTRAN 3.0
DEC DECwindows OSF/Motif 1.1.1 developers toolkit

d. SUN Solaris/SPARC Workstation

Solaris 1.0 (SunOS 4.1.1)
Sun C V1.1
Sun Fortran V1.4
OPENwindows V3.0
DEC DECwindows OSF/Motif 1.1.1 developers toolkit

SECTION III

COMPUTATIONAL COMPONENT IMPLEMENTATION

A. VERSION 1

1. Fuzzy c-Variety Clustering

The Fuzzy c-Variety Cluster analysis algorithm was implemented as a standalone component, *XFCV_Clustering*. This enables the user to perform the data analysis (which is computationally intensive) on a platform with adequate CPU performance and then visualize the resulting data on perhaps another platform, one more suited to the graphical component of the system. The XFCV_Clustering module was implemented using standard FORTRAN-77 utilizing no vendor specific extensions. This makes it possible to utilize the module on any platform supporting FORTRAN-77.

2. Principal Component Analysis

The Principal Components analysis component was implmented in a similar fashion to XFCV_Clustering. This component is referred to as *XFCV_PCPlot*.

B. VERSION 2

1. Fuzzy c-Variety Clustering

Various changes were made in the user interface to make it more stable (less prone to crashing due to user input errors). In addition, the format of the input data file and some output files were changed as noted later.

2. Principal Components Analysis

The user interface to this component was improved to make it more stable. Modifications were applied to support new data file formats for V2.

C. VERSION 3

1. Data point 'clipping'

This feature would allow the user to select particular data points to be omitted from the clustering and projection analysis (additional details can be

found later in the data editing section). In addition, the user will be able to specify various criteria for the system to use to clip certain data points itself.

2. Faster Eigen analysis

We anticipate replacing the outdated *Eigrs* routine currently used with a faster routine, which should alleviate some of the performance issues of the current version.

SECTION IV

GRAPHICAL COMPONENT IMPLEMENTATION

A. VERSION 1

1. DEC VAX/VMS Workstation

The initial version of the graphical component was developed on this platform due to the existence of good software tools for program development. The graphical component, hereafter referred to as *XFCV*, was developed using the C language. The original plan was to use GKS to implement the actual graphics rendering, but after some research it was determined that GKS lacked the user interface features required by a system of this type and GKS was not universally available across a wide variety of platforms. The decision was then made to utilize the X Window System for the user interface and graphics rendering. At the time of initial development, the GUI supported on this platform was the DECwindows GUI. DECwindows provided a stable toolkit with which to generate a user interface for the system. During the time of the first version, the Open Software Foundation was formed and defined a GUI standard (OSF/Motif), which was to be based partly on the DECwindows toolkit, enabling a reasonably easy task of later converting the system to OSF/Motif, which would then allow the system to be ported to a wider variety of platforms.

Version 1 of the system provided basic visualization capabilities, with only cursory support for viewing larger data sets.

2. Intel 80x86/MS-DOS Personal Computer

The port of the system to this platform utilized the Microsoft Windows GUI, which is shares much of the "look and feel" of OSF/Motif. This allows users to easily move between future versions of the workstation platforms (which will utilize OSF/Motif when it is generally available) and the personal computer platform. Initially impressions of this platform were that it was very slow with regards to the graphics performance. These issues could be addressed with faster hardware (refer to later section discussing hardware and software evaluations).

Although the user interface is very similar in appearance to the workstation version, the application programming interface (API) for Microsoft Windows in drastically different from an X Window based system. For this reason, the source code for the two platforms are maintained separately, with the Microsoft Windows version referred to as *MSFCV*.

3. DEC RISC/Ultrix Workstation

Support for this platform required few changes over the VAX/VMS version, as both platforms supported the DECwindows GUI. Specific changes dealt with the difference in the operating systems (OS) with regards to the execution of background jobs (needed to support external, standalone components of the system).

B. VERSION 2

1. Changes from Version 1

a. Print Screen

Version 1 of the system utilized an somewhat limited external program to perform the Print Screen function. Version 2 supports the use of *xgrabsc* (a public domain program) to perform the Print Screen function. The original public domain version of the user interface for this program was enhanced to use a OSF/Motif GUI to allow it to better integrate with other system components.

xgrabsc allows the user to print the entire visible display or a user selected portion of the display. The user can also choose various parameters for the print, including: save print to file or send to specified queue, color or greyscale PostScript, orientation (landscape or portrait mode), etc.

b. Extended on-line help

The on-line help system was revamped for V2 to enable the use of the Help Widget which is part of the DEC supplied OSF/Motif toolkit.

c. Transformations

The Transformations function in the system was extended to support real-time transformations as the user selects a rotation or zoom factor. Previously, transformations would be done after the user selected a specific value using the zoom/rotation sliders.

2. New Features

a. Viewport panning

This feature permits the user to view a data set which would otherwise not fit on the physical display. It also enable the user to zoom in on the data set and 'pan' around the display.

b. Data point editing

The system was extended to support a pop-up window which would allow the user to change various values associated with a data point, such as: x, y, y coordinates, class membership values, coloring values, cluster assignment, etc.

c. Motif support

As development of the original workstation progressed, it was determined that the choice of using the OSF/Motif GUI definition would significantly enhance the systems portability to a wide range of platforms. After specific additional functionality was added to the original DECwindows version, the entire system was converted to using the OSF/Motif GUI.

d. Axis display

Version 2 of the system was enhanced to display user selectable axis, in both the data set viewing area and also on the transformations pop-up window. The user can select between viewing the positive or negative axis, both axis, or no axis at all. In addition, the axis dynamically move as the orientation on the data set is changed during user selected transformations and data set animation.

e. Visualization enhancements

Various additions were made to the available visualization options including: fuzzy coloring/numbering, starburst cluster viewing, user assignable primary cluster definitions, etc. Refer to the *XFCV User's Manual* for complete details of these options.

3. Platform specific enhancements/restrictions

In general, the XFCV looks and behaves similarly across all supported platforms. The following are a few specific issues relating the implementation of the system on the particular platforms. For more information, refer to the *XFCV Release Notes V2.0.*

a. DEC VAX/VMS Workstation

There are some system initialization performance issues that were introduced following the utilization of the OSF/Motif GUI (due to its larger memory and system requirements compared to the DECwindows GUI) which need to be resolved for this platform. Once the system finishes initializing, it performs as expected. We expect the initialization performance to improve in the next release.

There are some issues with VMS data file formats which were not resolvable in this release. The issues relate to the internal VMS file/record formats and are not due to the way XFCV system writes data. Refer to the *XFCV User's Manual* for more information for handling various data file format issues under VMS.

b. Intel 80x86/MS-DOS Personal Computer

The *Print Screen* function is not currently supported in this release for the MS-DOS/Windows platform.

The performance of this platform (80286-based) are not considered adequate to perform data analysis or graphical visualization. See additional notes in the Hardware/Software evaluation section under Results and Conclusions.

c. SUN Solaris/SPARC Workstation

This platform was certified for the V2.0 release (using DEC's OSF/Motif V1.1 toolkit). Although OSF/Motif is available from other vendors for the SPARCstation, the system currently depends on the Help Widget supplied by DEC as part of their OSF/Motif offering. We expect this limitation to be removed once OSF supports a Help Widget as part of the standard Motif distribution.

C. VERSION 3

1. Future Plans

The following details various plans for future development of the system. Some enhancements are items which were not completely developed in time for the V2 release; others are new concepts.

a. Volume visualization

(1) Wireframes

Much research was done during the first two versions of the graphical component to develop algorithms to allow the clusters to be viewed as wire frame models. The wireframe modeling would allow the user to gain information of the relative spread of the cluster members by visualizing the volume of space which the cluster occupies. The research to implement this was not complete for Version 2 of the system.

(2) Solids modeling

In addition to the research that went into the wireframe modeling, additional work was done on the concept of applying solids modeling to the cluster

visualization. By displaying the data clusters as shaded volumes, the user would not only gain insight into the spread of the data by the volume of space which is displaced, but the addition of shaded surfaces may aid the user in determining additional information regarding the higher dimensionality of the data which is otherwise lost by the n-dimension to three-dimension projection process.

(3) Transparent 'cloud'

This method is a variant on the solids modeling concept. Basically it would allow not only the exterior data points in a cluster to be observed, but also the "interior" points.

b. Cluster clipping

After using the system, it has been determined that upon occasion it is useful to be able to omit a particular cluster from the visual display, in order to concentrate more on the interaction between other clusters. This enhancement would allow the user to selectively remove a particular cluster either from the analysis or visualization stage of the system. Currently, this can only be done by editing the raw data with the spreadsheet to remove the unwanted cluster(s) and then reperform all analysis and visualization.

c. Data point clipping

Often during the process of visualizing a data set, one determines that a particular data point (or set of points) should be removed in order to concentrate on other data points. These points could be considered outliers or merely "noise" by the user and need to be selectively removed either temporarily or permanently from the data set. Currently, the only way to achieve this effect would be to use the spreadsheet to remove the unwanted point(s) and the reperform all analysis and visualization, which may not have the desired effect (as all points in a data set contribute to the projection of it and by removing some points a new projection will result, which may not be the effect the user is trying to achieve).

d. Context sensitive on-line help

Although the current on-line help is considered useful, it would be much more usable if the user to use the mouse to select an item on which to obtain help, rather than traversing the tree structure of the help system. This would also lead to a better answer for the user, as the help system would be made to understand the current "mode" of the item the user is selecting, thus it could limit the help information given to exactly what is applicable rather than a more broad piece of information (as is now given).

e. Better data set management

This is an addition that overlaps with the data editing component of the system. After use of the system, it has become apparent that the "moded" approach (in which the user uses a spreadsheet to edit the raw data, but then uses an integral part of the graphical system to modify processed data) could be improved. What should be done is to allow the spreadsheet to more easily process any/all data which needs modification. This would allow the user to utilize the same component for all data manipulation tasks rather than several.

2. Platform support

In addition to specific feature enhancement, the following details enhancements relating specifically to platform issues (hardware and software).

a. Open Look GUI support

Using the Open Look GUI specification will allow the system to make the best use of the SUN SPARCstation platform. Although the OSF/Motif based version of the graphical component functions well on the SPARCstation platform, Sun does not support OSF/Motif, instead choosing to supply their own GUI, based on the OPENLOOK definition. Thus, the stock SPARCstation platform is more tuned towards to the use the Sun supplied GUI (OpenWindows).

b. GL or PHIGS/PEXlib support

Using GL and/or PHIGS/PEXlib will allow the system to make use of 3D graphics accelerators on various platforms. Currently, windowing systems are 2D environments and XFCV perform its own 3D transformations. By the use of GL or PHIGS/PEXlib, the system could take advantage of the much more extensive 3D support provided by these libraries.

c. IBM RS/6000 series, HP 9000/700 series, SGI 4D series

All of these platforms are popular choices for graphical applications; the IBM and HP platforms have especially attractive CPU performance, relative to Sun and DEC systems. A number of these platforms also provide high-performance graphics hardware, which would further enhance the performance of the system.

d. 24 plane graphics adapter support

24 plane graphics adapters will allow better color depth to be used to show minute color differences between data points, as well as enabling smooth shading of the added visualization models. In addition, 24 color planes will allow larger data sets to be visualized, with all points assigned a distinct color. 24 color

planes theoretically would allow 2^{24} data samples to be viewed simulataneously, although the actual number of data samples would be limited by the resolution of the display hardware (the average display has far less than 2^{24} pixels).

e. Enhanced 8 plane graphics adapter support

Various 8 plane graphic adapters have varying capabilities relating to numbers of available color maps, double-buffering (which aids in the animation of the data sets), etc. The system should be modified to be able to use the extra functionality of these adapters where possible. In addition, the system currently limits the number of data samples that can be viewed simultaneously to 255 on 8 plane adapters (this is because an 8 plane adapter can only simultaneously display up to 255 distinct colors). It should be possible to allow larger data sets to be viewed by quantizing the color map required and using the closest available color for a data sample when the color map is full, rather than requiring that point to have its own distinct color. It has been observed that many times, the difference in color of many points is very minute and allowing points whose colors are "close" to share a single color would not seriously degrade the visualization of the data set.

SECTION V

MISCELLANEOUS COMPONENT IMPLEMENTATION

A. VERSION 1

1. Data Management

a. Data entry/editing

A stand-alone component was developed to handle the task of creating and editing data sets, *XFCV_INPUT*. The program was designed to give the user the ability to enter and edit data samples, in vectors. It was determined that the component was adequate for small data sets, but would have to be modified in the next release to handle larger data sets.

b. Data scaling

A stand-alone component was developed to permit the user to impose various scaling operations (e.g., autoscale, log scale, normalization, etc.) on the raw sample data sets.

2. Data Format

a. Raw sample data

The raw sample data files contain data vectors of multivariate data samples. The system can handle data sets with up to fifty dimensions in this version. The format of the raw sample data was defined to be:

```
ns, nd
format
1   data-vector
2   data-vector
    .
    .
    .
ns  data-vector
```

where *ns* is the number of data samples, *nd* is the number of descriptors per data

sample, *format*, which is a FORTRAN-77 floating point format specification (e.g., F11.4) for each element of the data vectors in the file, the *1, 2 ...* are the sample #s (in a I3 format), and *data-vector*, which is the set of measurements (floating-point) comprising the data sample.

A sample raw data set would look like:

```
10,  4
F11.4
  1    50.0000    33.0000    14.0000    2.0000
  2    46.0000    34.0000    14.0000    3.0000
  3    46.0000    36.0000    10.0000    2.0000
  4    51.0000    33.0000    17.0000    5.0000
  5    55.0000    35.0000    13.0000    2.0000
  6    48.0000    31.0000    16.0000    2.0000
  7    52.0000    34.0000    14.0000    2.0000
  8    49.0000    36.0000    14.0000    1.0000
  9    44.0000    32.0000    13.0000    2.0000
 10    50.0000    35.0000    16.0000    6.0000
```

b. PC plot data

The PC Plot data file defines the (x, y, z) coordinates of each data sample which is used to project the sample on the display. The PC Plot data file is created by the PC Plot component. In general, it does not need to be modified by the user outside of the XFCV system (i.e., with a text editor) and its format may be subject to change in future releases of XFCV. The specific format used by Version 1.x was:

```
ns
x1      y1      z1
x2      y2      z2
        .
        .
        .
xns     yns     zns
```

where *ns* is the number of data samples, and *x, y, z* are the coordinates of each data sample. An example of a V1.x PC plot data file:

```
6
-1.2     2.0     -0.3
-0.9     1.5     -0.1
 2.0     5.4      0.5
 2.5     4.5      0.1
 4.3     9.2     -0.9
 5.2     7.4     -1.3
```

where the first line specifies the number of samples, and the lines which follow list the *x, y, z* coordinates for each sample point in the PC plot.

c. Class Membership

The Class Membership information is the resulting output of the *XFCV_Clustering* component. In general the class membership data would not need to be modified outside of the XFCV system and may be subject to change in future releases. For each cluster in the data set, the data file contains the specific membership value for that cluster for each data sample. The specific data file format for Version 1.x was:

ns, nc

$$\begin{matrix} c_{1,1} & c_{1,2} & \cdots & c_{1,nc} \\ & & \vdots & \\ c_{ns,1} & c_{ns,2} & \cdots & c_{ns,nc} \end{matrix}$$

where *ns* is the number of data samples, *nc* is the number of cluster (classes), and $c_{n,m}$ is the specific class membership value for class *m* of sample number *n*. The coloring of the data in the graphical component (XFCV) is derived from the class membership data. An example of a V1.x class membership data file would look like:

```
6   3
1   3   6
.92 .07 .01
.89 .08 .03
.04 .94 .02
.05 .93 .02
.02 .02 .96
.01 .02 .97
```

where the first line specifies the number of samples and classes, the second line lists the cluster centers of each class, and the lines following that list the class membership values for each sample. The coloring of the data in the graphical component (XFCV) is derived from the class membership data.

d. RGB data

The RGB data file is generally created by the *XFCV/MSFCV* component. It allows the user to save the specific coloring of a data set for later display. In

addition to the coloring information, the RGB file also contains the specific cluster assignments made (if any) during the process of visualizing the data set. The RGB data file is used internally in the XFCV system and is generally not modified outside of the context of the system; its format is subject to change in future releases of the XFCV system. The specific format of the V1.x RGB data file was:

ns
r_1 g_1 b_1 c_1
.
.
.
r_{ns} g_{ns} b_{ns} c_{ns}

where *ns* is the number of data samples and r_i, g_i, b_i, c_i are the red, green, blue color definitions and cluster assignment for data sample number i respectively. An example RGB data file:

```
6
235     18      2
227     20      8
 10    240      5
 13    237      5
  5      5    245
  2      5    247
```

where the first line specifies the number of samples to follow, and the remaining lines define the *RGB* coloring for each sample.

B. VERSION 2

1. Changes from Version 1

a. Data Formats

(1) Raw sample data

The format of the raw sample data was changed to eliminate the requirement to include the data format specifier (e.g., Fm.n where *m* is the width of each sample number and *n* in the number of digits after the "."). It was determined that this would make it easier for the system to accept data from various sources, where it is not always possible for the data to be obtained in such a restrictive fixed format. In addition, the dimensionality of the data samples was extended. It is now possible to define data samples of up to 100 dimensions. All fields are now unformatted and separated by a comma (,).

The V2.x raw sample format is:

ns, nd
1, data-vector
2, data-vector
.
.
.
ns, data-vector

where *ns* is the number of data samples, *nd* is the number of descriptors per data sample, the *1, 2* ... are the sample #s, and *data-vector*, which is the set of measurements (floating-point) comprising the data sample.

In the new format, a sample data file would look like:

10, 4
1, 50.0, 33.0, 14.0, 2.0
2, 46.0, 34.0, 14.0, 3.0
3, 46.0, 36.0, 10.0, 2.0
4, 51.0, 33.0, 17.0, 5.0
5, 55.0, 35.0, 13.0, 2.0
6, 48.0, 31.0, 16.0, 2.0
7, 52.0, 34.0, 14.0, 2.0
8, 49.0, 36.0, 14.0, 1.0
9, 44.0, 32.0, 13.0, 2.0
10, 50.0, 35.0, 16.0, 6.0

(2) Class Membership data

The Class Membership data was extended to include the cluster center information chosen by the user at the time of the clustering analysis. This makes it possible for *XFCV* to display the cluster center among the group of data points in each cluster, in addition to enabling the starburst visualization option.

(3) RGB data

The RGB color data file format was modified to remove the user-defined primary class membership, as it was determined that this information belongs in its own data file, to facilitate loading/storing/modifying that information independent of the RGB data.

b. Data scaling/editing components

The data scaling and editing components were updated to accomodate the changes in data formats. In addition, various bugs in the user interface of the scaling component were fixed. See notes below on changes to the data editing component.

2. New Features

The data entry/editing component (XFCV_INPUT) that was developed for Version 1 was replaced. It was determined that the component was inadequate for the future needs of the system and it would require much additional work to enhance it. On the VAX/VMS and Unix platforms XFCV_INPUT was replaced with the public-domain spreadsheet *sc*. The *sc* spreadsheet allows for data entry and editing of the large data sets required for the XFCV system. In addition, because the source code was available, it could be modified easily to meet the particular needs of the system. On both workstation platforms *sc* was modified to read and write the particular data formats for the various XFCV components.

On the Intel 80x86/MS-DOS Personal Computer platform, it was determined that the best solution was to use a readily available commercial spreadsheet (e.g., Lotus 1-2-3, Quattro-Pro, etc). The system was enhanced by the addition of some data conversion routines to convert plain text data files to the format required by the commercial spreadsheet (in this case, DIF).

C. VERSION 3

Integrate spreadsheet/scaling components to allow for better overall data set management. The intention is to bring the functionality of all data modification/transformation into a single module instead of two smaller, discrete modules. In addition, some extensions should be made to enable better data set management, via the use of a database of some form. This will enhance the overall system by making it easier to manage large numbers of data sets. It may be possible to use a commercial spreadsheet/database solution for all platforms (VMS, Unix and MS-DOS) in order to reduce the amount of effort spent duplicating functionality already in existence (in commercial products).

SECTION VI

RESULTS, CONCLUSIONS, AND RECOMMENDATIONS

A. RESULTS

1. Neat Fuel Data Set

The test data consisted of 201 samples representing five different types of jet fuels (see Table 1) obtained from Wright Patterson Air Force Base and Mukilteo WA Energy Management Laboratory. These two laboratories examine batches of fuels purchased for use by the Department of Defense to verify that the purchased fuels match designated specifications. Splits from regular quality control samples were collected over a period of time and constituted a representative sampling of the fuels.

TABLE 1. NEAT JET FUEL DATA SET

Fuel Type	Number of Samples
JP-4 (Fuel used by United States Air Force)	97
Jet-A (Fuel used by Civilian Airliners)	52
JP-7 (Fuel used by the SR-71 Aircraft)	15
JPTS (Fuel used by the U-2 and TR-1 Aircraft)	14
JP-5 (Fuel used by the Navy)	23
Total	201

Five types of neat jet fuels were studied by GC/MS: JP-4, Jet-A, JP-7, JPTS, and JP-5. Prior to GC/MS analysis, each fuel sample was diluted with methylene

chloride and spiked with d_{10}-anthracene which served as an internal standard. The diluted fuel sample was then injected directly onto a capillary column temperature programmed from 40°C to 250°C at 3°C/minute, and the mass spectrometer was scanned cyclically over the mass range 25 to 350 daltons. The data were collected and stored on an HP-1000-F minicomputer. The GC/MS data for the study was collected by Dr. Howard Mayfield of the Engineering & Services Laboratory, Tyndall Air Force Base, and full details about the data collection are given elsewhere (19).

The total ion chromatograms were peak matched using a computer program developed by Mayfield and Bertsch (20) which proceeds from reference peaks identified by the user as markers to force time-base consistency on individual runs to ensure best fit with respect to a reference chromatogram. The computer program used for peak matching yielded a set of 76 standardized retention time windows.

Each peak matched chromatogram was initially represented as a data vector $x = (x_1, x_2, x_3, \ldots\ldots x_j, \ldots\ldots x_n)$ where x_j is the area of the jth peak normalized so using the d_{10}-anthracene peak. The set of data -- 201 chromatograms of 76 peaks each -- were autoscaled to remove any bias arising from differences in magnitude so that all chromatograms and features have equal weight in the analysis.

In this study FCV-false color data imaging was used to investigate the structure of the data. The starting centers for the FCV-false color data imaging experiments were determined directly by the user. In these imaging experiments, r was set equal to zero; m was set equal to 2; and the minimum prespecified change criterion for the class membership value was set equal to 0.005. (Our experience with GC profile data has been that r should be set to zero.) 63 of the 76 standardized retention time windows were retained as descriptors for these imaging experiments. The 63 peaks chosen possessed a common set of attributes: (1) each peak was well resolved and readily identifiable in all of the chromatograms; (2) each peak had a reliable area count; and (3) each peak conveyed unique information about the classification problem at hand, i.e., the correlation coefficient between any two peaks of the 63 selected was less than 0.98.

Figure 3 shows a 2-dimensional FCV-false color data imaging print of the Jet-A fuel data. (The number of clusters sought, c, in the data was 3 which is equal to the number of suspected Jet-A sub-classes; the colors assigned to the clusters were red, blue, and green.) The 1s represent jet fuel samples obtained from Wright Patterson Air Force Base; the 2s represent jet fuel samples obtained from Mukilteo. The 1s and the 2s were analyzed at the beginning of the study on different days, i.e., the 1s were run on one day and the 2s were run on another day. The 3s represent chromatograms run near the end of the study when stricter quality control standards were employed, e.g., a sample would be re-analyzed if the internal standard did not appear as a well defined peak in the chromatogram. Jet-A fuel samples comprising this group were obtained from both Wright Patterson and Mukilteo.

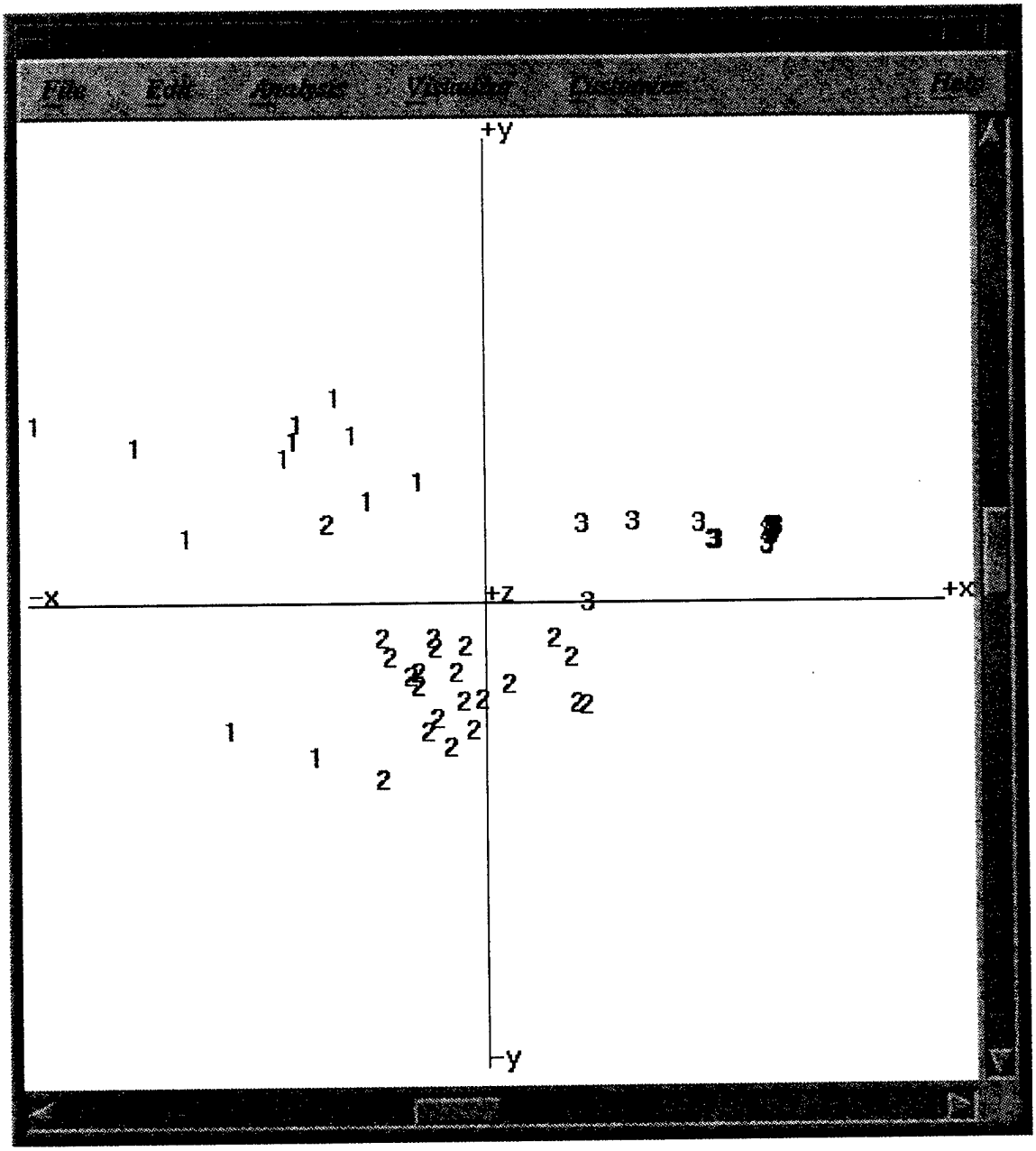


Figure 3. A 2-Dimensional False Color Data Image Print of the Jet-A Fuel Chromatograms.

The spatial distribution of the data points in the map suggests that instrumental drift, presumably due to changes in the operating conditions of the DB-5 column, is a serious problem. (The class designation 1, 2, and 3 for the Jet-A samples also denotes the time period when the analysis was performed; note that the 1s, 2s, and 3s cluster in different regions of the sub-space.) Since principal component analysis does not use class information about individual samples, the apparent separation would be considered a strong indication of real differences in the chromatograms due to changes in the operating conditions. However, the 1s and the 2s have a similar color, i.e., these samples have a color that is a mixture of both red and green, which suggests considerable overlap between the two sets of data points. (If these sets of points do not lie close to one another in the high-dimensional space, they would be displayed in different colors.)

To better understand the structure of this data space, further false color data imaging experiments were performed. Figure 4 shows a 3-dimensional FCV false color data imaging print with rotation of the same Jet-A data set. (Again, c was equal to 3 and the colors red, blue, and green were used to represent each of the clusters.) The location of the sample points in the subspace is consistent with their color. There is considerable overlap between the 1s and the 2s in the subspace defined by the three largest principal components. The 3s, on the other hand, form a compact cluster of points surrounded by both 1s and the 2s. The fact that the 3s are well separated from the 1s and the 2s and have a very different color is not surprising in view of the fact that these chromatograms were run under more stringent operating conditions. It is also interesting to note that principal component analysis had actually exaggerated differences in the concentration patterns between chromatograms during the projection process (see Figure 3). Through the use of color which was provided by the membership coefficients of the FCV clustering algorithm, spatial relationships between individual sets of points in the high dimensional measurement space were restored.

Figure 5 summarizes the results of a false color data imaging experiment for the JP-7 fuel samples. (The number of clusters, c, was set equal to two, and the colors red and green were used to represent the clusters.) The 1s represent fuel samples obtained from Mukilteo, and the 2s represent fuel samples obtained from both Wright Patterson and Mukilteo. The 1s were analyzed via GC/MS at the beginning of the study, whereas the 2s were run at the end of the study when stricter quality control standards were used for rejection of data. The fact that the 1s are well separated from the 2s on the map and also have a very different color suggests that instrumental drift is a serious problem with this set of data.

False color data imaging experiments were also performed on the JP-4, JPTS, and JP-5 fuels. Instrumental drift also appeared to be a serious problem in each of these fuel data sets. In other words, clustering of the data points on the basis of this unwanted experimental effect was observed (in varying degrees) in false color data imaging maps of the JP-4, JPTS, and JP-5 data.

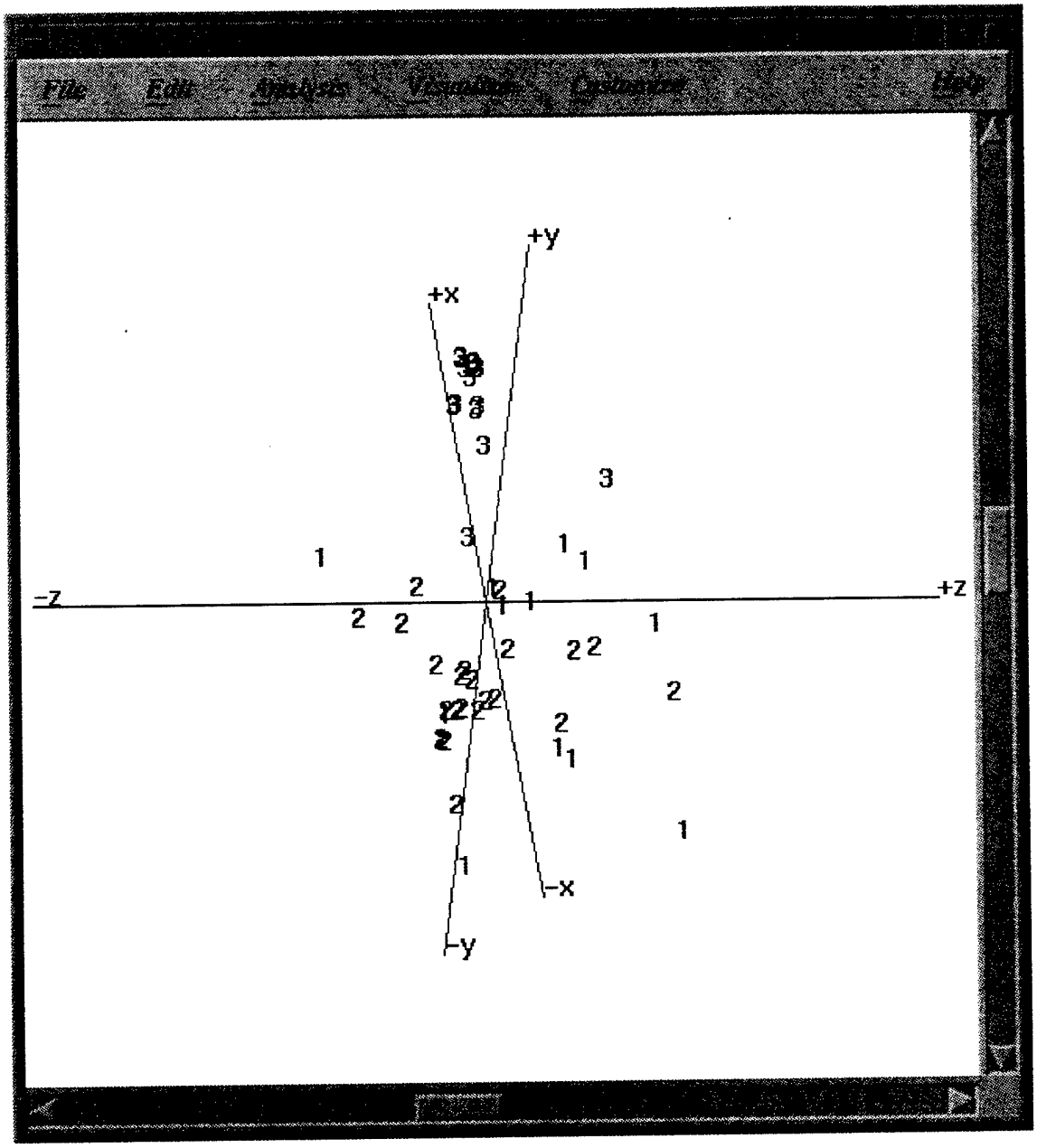


Figure 4. A 3-Dimensional False Color Data Image Print with Rotation of the Jet-A Fuel Chromatograms.

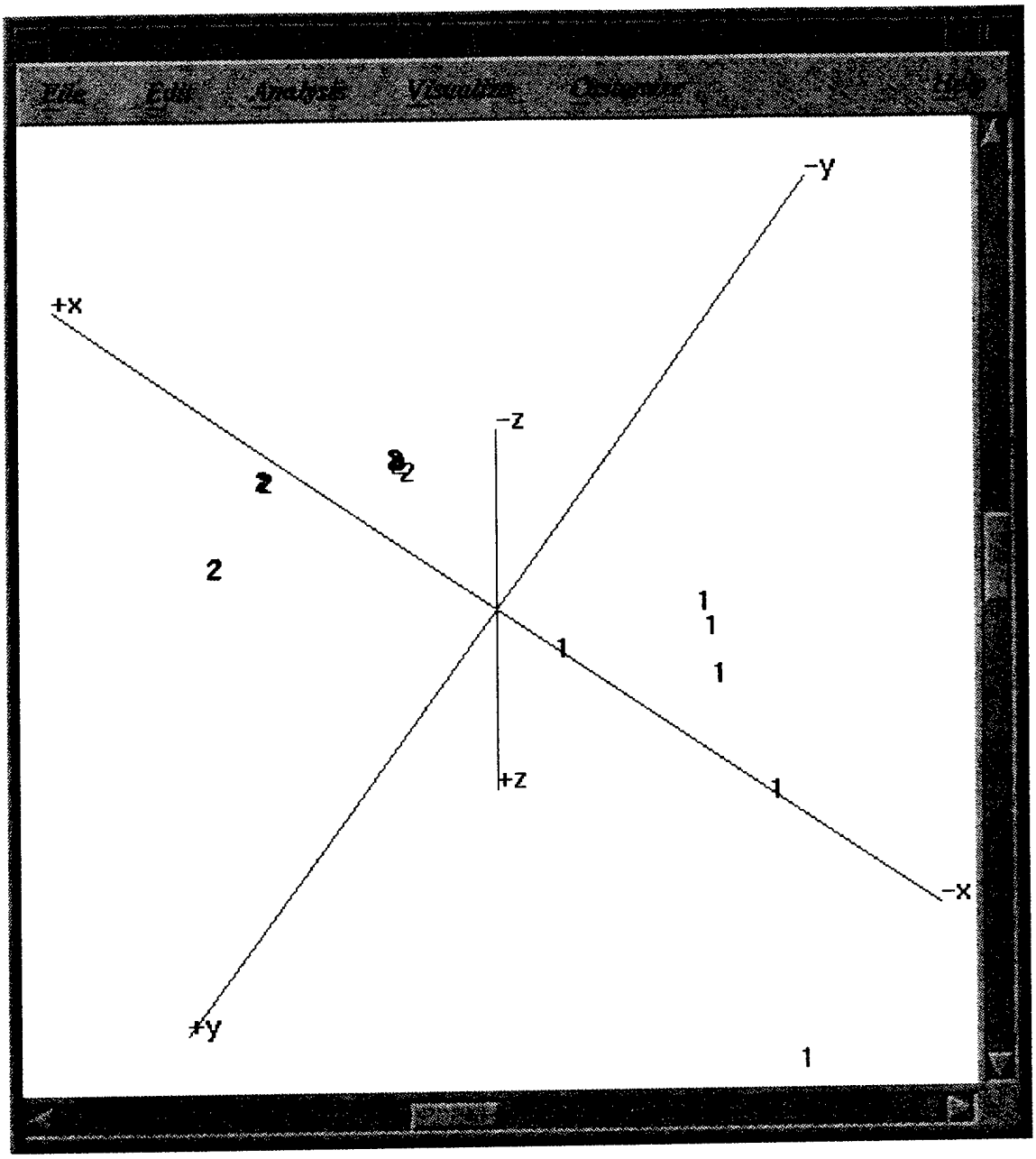


Figure 5. A 3-Dimensional False Color Data Image Print with Rotation of the JP-7 Chromatograms.

Having studied the structure of the data with mapping and display techniques, the next step was to assess the effects of instrumental drift and sample variability on classification. Principal component models for each fuel type were developed using SIMCA. The complexity of each model was estimated directly from the data using the technique of cross validation (21). The chromatograms in the training set were fitted to the class models, and classifications were made according to the goodness of fits using either the standard deviation criterion or the uniqueness criterion (12). The results of this study are summarized in Table 2. They show that JP-7 and JP-4 can be readily differentiated from the other fuels. However, Jet-A, JP-5, and JPTS cannot be identified reliably from GC profile data.

TABLE 2. SIMCA CLASSIFICATION RESULTS

sd-criterion

Class	Number of PC's	NIC	Right	Wrong	PCT Right
JP-4	5	97	95	2	97.9
Jet-A	1	52	28	24	53.9
JP-7	5	15	15	0	100.0
JPTS	3	14	9	5	64.3
JP-5	4	23	16	7	69.6
Total		201	163	38	81.09

TABLE 2. SIMCA CLASSIFICATION RESULTS (CONCLUDED)

uniqueness criterion

Class	Number of PC's	NIC	Right	Wrong	PCT Right
JP-4	5	97	93	4	95.9
Jet-A	1	52	13	39	25.0
JP-7	5	15	15	0	100.0
JPTS	3	14	7	7	50.0
JP-5	4	23	14	9	61.0
Total		201	130	71	64.6

A five-way classification study involving only the jet fuel samples that were run when stricter standards were used for rejection of gas chromatographic data was also undertaken. In essence, the factor in the data associated with instrumental drift has been blocked out by using only these samples in the training set. Again, principal component models for each fuel type in the training set were developed using SIMCA, and classifications were made according to the goodness of fit statistic using either the sd-criterion or the uniqueness criterion. The classification of the 67 jet fuel samples was perfect (see Table 3). Evidently, instrumental drift due to changes in the operating conditions of the GC column can be a serious problem which must be taken into account when designing experiments of the type reported here. On the other hand, sample variablility attributed to the different manufacturing processes used to generate a particular jet fuel type does not affect the classification of neat jet fuels. (Recall that the jet fuel samples constituting the 3s were obtained either from Wright Patterson or Mukilteo and probably constitute a representative sampling.)

TABLE 3. SIMCA CLASSIFICATION RESULTS

sd-criterion

Class	Number of PC's	NIC	Right	Wrong	PCT Right
JP-4	5	33	33	0	100.0
Jet-A	3	14	14	0	100.0
JP-7	1	6	6	0	100.0
JPTS	1	6	6	0	100.0
JP-5	2	8	8	0	100.0
Total		67	67	0	100.0

uniqueness criterion

Class	Number of PC's	NIC	Right	Wrong	PCT Right
JP-4	5	33	33	0	100.0
Jet-A	3	14	14	0	100.0
JP-7	1	6	6	0	100.0
JPTS	1	6	6	0	100.0
JP-5	2	8	8	0	100.0
Total		67	67	0	100.0

2. The Mixed Fuel Data Set

The test data consisted of 95 total ion chromatograms of five types of neat jet fuels, as well as binary mixtures of the fuels (i.e., 50/50 mixtures). Dr. Howard Mayfield of the Engineering & Services Laboratory, Tyndall Air Force Base, FL collected the GC/MS data, and full details about the collection of the data can be found elsewhere (19). The 95 total ion chromatograms were peak matched using a computer program developed by Mayfield and Bertsch (20). The computer program used for peak matching yielded a set of 76 standardized retention time windows. (In fact, the data was transduced using the same 76 feature retention time template as was used for the neat fuel data described above.) The chromatographic data - 95 chromatograms of 76 peaks each - were normalized to constant sum using the total integrated peak area and autoscaled to ensure that all chromatograms and features have equal weight in the analysis. The 95 chromatogram data set was divided into a training set of 80 chromatograms and a prediction set of 15 chromatograms (see Table 4). The training set consisted of the neat jet fuels, and the prediction set was comprised of the mixtures of these neat jet fuels.

TABLE 4. MIXED FUEL DATA SET

Fuel Type	Number of Samples
JP-4	32
Jet-A	15
JP-7	7
JPTS	7
AVGAS	19
Mixtures	15
Total	95

Principal component analysis was used to study the data. Figure 6 shows a principal component map of the training set data. The chromatograms for the fuel samples are represented as points in the 2-dimensional map. Clustering of the points according to fuel type is observed for this data suggesting that information about fuel type is present within the chromatograms.

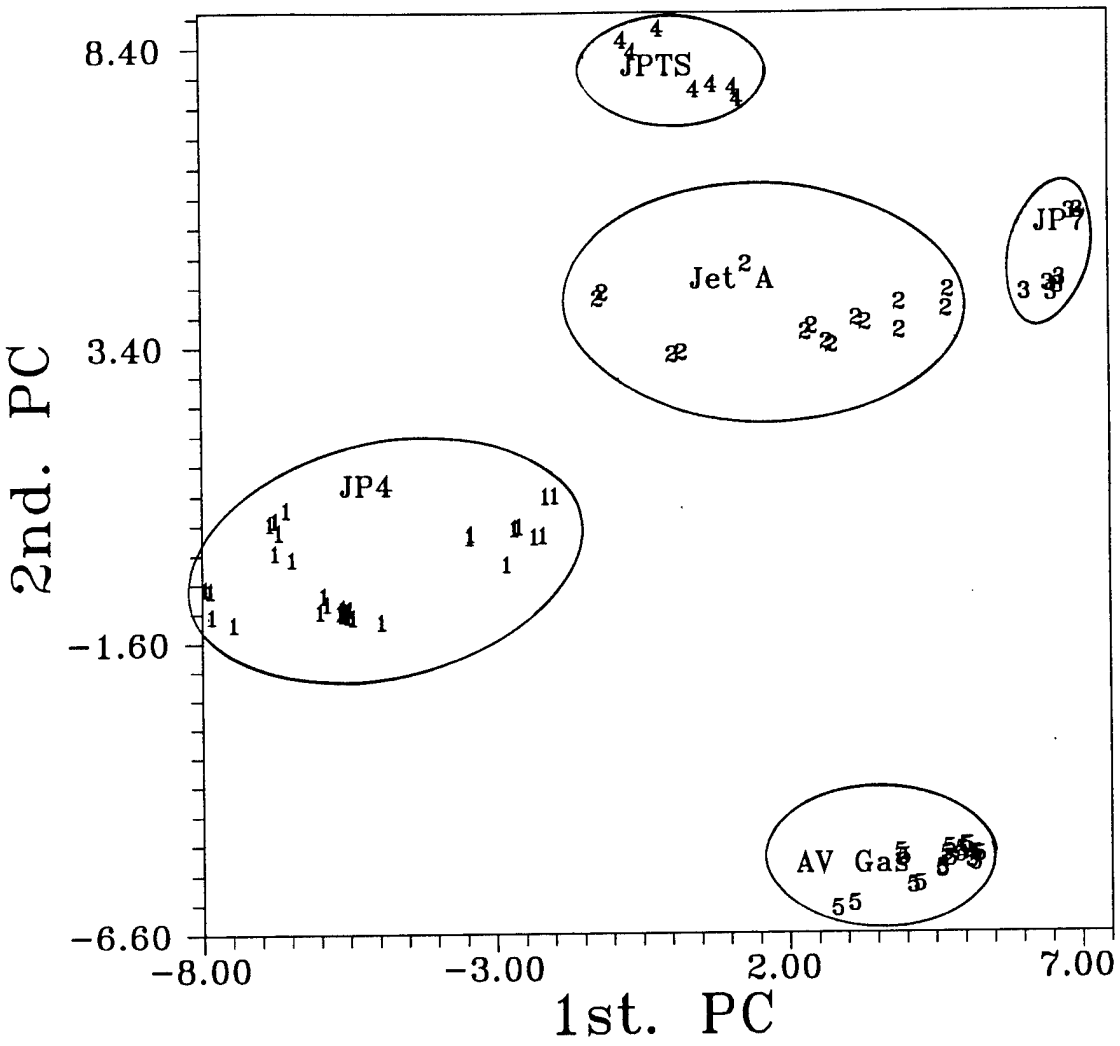


Figure 6. A Plot of the Two Largest Principal Components of the 58 GC Peaks for the Training Set Data. The Two Largest Principal Components Explain 80 percent of the Total Cumulative Variance in the Data. 1 = JP-4; 2 = Jet-A; 3 = JP-7; 4 = JPTS; and 5 = AVGAS.

The FCV clustering algorithm was also used to study the structure of the data. GC peaks present in at least 10 percent of the chromatograms (i.e., 58 of the 75 GC peaks) were used as a starting point for the analysis. The starting centers for the cluster analysis were supplied by the user and were prototypical vectors representative of each fuel type in the data. The minimum prespecified change criterion for the class membership value was set equal to 0.005; r was set equal to zero; and m was set equal to 2. As c increased in value, clustering of the samples on

the basis of fuel type was observed. When c = 6, an interesting result was obtained. 23 of the 32 JP-4 samples had a large class membership value (greater than 0.75) for cluster number one. None of the other samples had a large class membership value for that cluster. The other 9 JP-4 samples had a large class membership value for cluster number two. None of the other samples had a large class membership value for that cluster. All of the Jet-A samples had a large class membership value for cluster number three. Again, none of the other samples had a large class membership value for cluster number three. All of the JP-7 samples had a large class membership value for cluster number four, and all of the JPTS samples had a large class membership value for cluster number five. Again, no other samples had a large class membership value for these clusters four or five. The AVGAS samples and only the AVGAS samples had a large class membership value for cluster number six. When the fuel samples were assigned to the cluster where they had the highest class membership value, it was evident the data could be partitioned into different classes on the basis of fuel type.

TABLE 5. PREDICATION SET RESULTS

Sample	Components	FCV Prediction
1	JP-4/Jet-A	JP-4/Jet-A
2	JP-4/JP-5	JP-4
3	JP-4/JP-7	Jet-A
4	JP-4/JPTS	JP-4/JPTS
5	JP-4/AVGAS	JP-4/AVGAS
6	JP-5/JP-7	Jet-A/JP-7
7	JP-5/JPTS	JPTS
8	Jet-A/JP-5	Jet-A/JPTS
9	AVGAS/JP-5	AVGAS/Jet-A
10	JPTS/JP-7	JPTS/Jet-A
11	Jet-A/JP-7	Jet-A/JP-7
12	AVGAS/JP-7	Jet-A/JP-7
13	Jet-A/JPTS	Jet-A/JP-7
14	AVGAS/JPTS	JPTS
15	AVGAS/Jet-A	Jet-A

The six principal component models developed in this experiment were tested using the 15 samples in the prediction set. The samples were fitted to each principal component model, and the class membership values for each sample were computed. Each sample was assigned to the cluster where it had a large membership value. Four of the fifteen binary mixtures were correctly classified using this procedure, i.e.,

the PC models were able to identify both fuel components present in each of these four samples. Nine binary mixtures were only partially classified, that is the six PC models were able to identify correctly one of the fuel components in each of these mixtures. Since five of the nine binary mixtures contained JP-5 fuel which was not represented in the training set, the high rate of partial classifications is not unexpected. However, the PC models were not able to correctly identify any of the components in two of the mixtures. The results of this study are summarized in Table 5 and demonstrate the feasibility of performing linear mixture analysis using GC data.

3. Water Soluble Data Set

This study focused on water samples that contain dissolved hydrocarbons, i.e., water that has been contaminated by neat jet fuels. The object was to determine whether or not total ion chromatograms of these dissolved hydrocarbons can be used to determine the particular type of jet fuel responsible for the contamination. Neat fuel samples of JP-4, Jet-A, JP-7, JPTS, 100/130 octane aviation gasoline (AVGAS) and diesel were obtained from either Wright-Patterson or Mukilteo Energy Management Laboratories. Water samples simulating ground-water contaminated by a jet fuel were prepared by equilibrating a neat jet fuel sample with Milli-Q water in a vessel designed by McIntyre (22) to maximize surface contact between fuel and water while avoiding mixing.

A simulated water sample was prepared by equilibrating two milliliters of fuel with 250 ml of water for 12 hours while stirring gently. Following equilibration, several milliliters of water were discharged from the vessel to ensure the delivery tube was clear of fuel, and two 25 ml aliquots of the water phase were delivered into gas tight syringes equipped with luer-lock open-shut valves. (Hence, two 25 ml water samples were prepared from a single fuel sample.) Each 25 ml aliquot was spiked with a solution of d_{10}-ethylbenzene in methanol and then forced through a C-18 Sep-pak (Millipore Corporation) solid phase extraction cartridge. The Sep-Pak was partially dried with a five milliliter slug of air and was extracted with 1 ml of carbon disulfide.

A one microliter aliquot of each carbon disulfide extract was injected directly onto the fused silica capillary column (60 meters long with an id of 0.25 mm) that was temperature programmed from 40°C to 200°C at 5°C per minute. The capillary column was coated with a 0.25 micron bonded polyethylene glycol stationary phase (DBWAX, J&W Scientific Inc). This high polarity stationary phase was chosen for these analyses because of its selectivity towards alkylbenzenes which are the major components found in the water soluble fraction of jet fuels (22).

Total ion chromatograms of the carbon disulfide extract were obtained using an HP 5987 GC/MS with an HP-1000-F minicomputer running the RTE-6/VM

operating system and the RTE-6/VM GC/MS Data System software. The manufacture's supplied software was used to subtract the mass 76 ion chromatogram from the total ion chromatogram of each sample in order to minimize the effect of carbon disulfide solvent on the sample chromatogram. The GC/MS data was collected by Dr. Howard Mayfield of the Engineering & Services Laboratory, Tyndall Air Force Base, and full details about the data collection are given elsewhere (19).

The GC profiles of the dissolved hydrocarbons were standardized using a computer program that correctly assigned peaks by dividing each chromatogram into intervals defined by major peaks that are always present (e.g., the d_{10}-ethylbenzene peak) and linearly scaling the retention times of the peaks within the intervals for best fit with respect to a reference chromatogram. The computer program used for standardization was developed by Mayfield and Bertsch, and full details about the program are given elsewhere (20). The peak-matching procedure yielded a set of 48 standardized retention time windows.

Each peak matched chromatogram was initially represented as a data vector $X = (x_1, x_2, x_3, x_j, \ldots\ldots x_{48})$ where component x_{48} is the area of the 48th GC peak. The chromatographic data -- 94 chromatograms of 48 peaks each -- were normalized to constant sum and also autoscaled to ensure that each feature and chromatogram had equal weight in the analysis.

Of the 94 chromatograms, 86 comprised the training set and eight constituted the predition set. The samples that constituted the prediction set were chosen by random lot. The types of samples that are represented in the data set are listed in Table 6.

TABLE 6. THE WATER SOLUBLE DATA SET

Training Set

Fuel Type	Number of Contaminated Water Samples	Number of Chromatograms
JP-4	11	22
Jet-A	13	26
JPTS	7	14
Diesel	5	10
JP-7	4	8
AVGAS	3	6
Total	43	86

TABLE 6. THE WATER SOLUBLE DATA SET (CONCLUDED)

Prediction Set

Fuel Type	Number of Contaminated Water Samples	Number of Chromatograms
JP-4	2	2
Jet-A	1	2
JPTS	1	2
Diesel	2	2
Total	6	8

The average of the replicate chromatograms was not used in the pattern recogntion analysis, since considerable information on the correlation structure between variables would be lost by combining the chromatograms. Instead, each ion chromatogram was treated as a unique object even though the number of independent samples in the data set is smaller than the number of data vectors, a fact of some importance when selecting variables on the basis of class separation.

The pattern recognition analyses in this study were directed towards three specific goals: (1) developing classifiers that could separate JP-4 contaminated water samples from water samples contaminated by other jet fuels, (2) studying the structure of the GC data to seek obscure relationships with mapping and display techniques, and (3) developing the ability to predict the class membership of unknowns.

For experiments of the type that we are considering, it is inevitable that relationships will exist among sets of conditions used in generating the data and the patterns that result. One must realize this in advance when approaching the task of analyzing such data. Therefore, the problem is to utilize the information in the chromatograms that is characteristic of the differences between the various jet fuels without being swamped by the large amounts of quantitative data due to experimental conditions that is also contained in the complex capillary chromatograms.

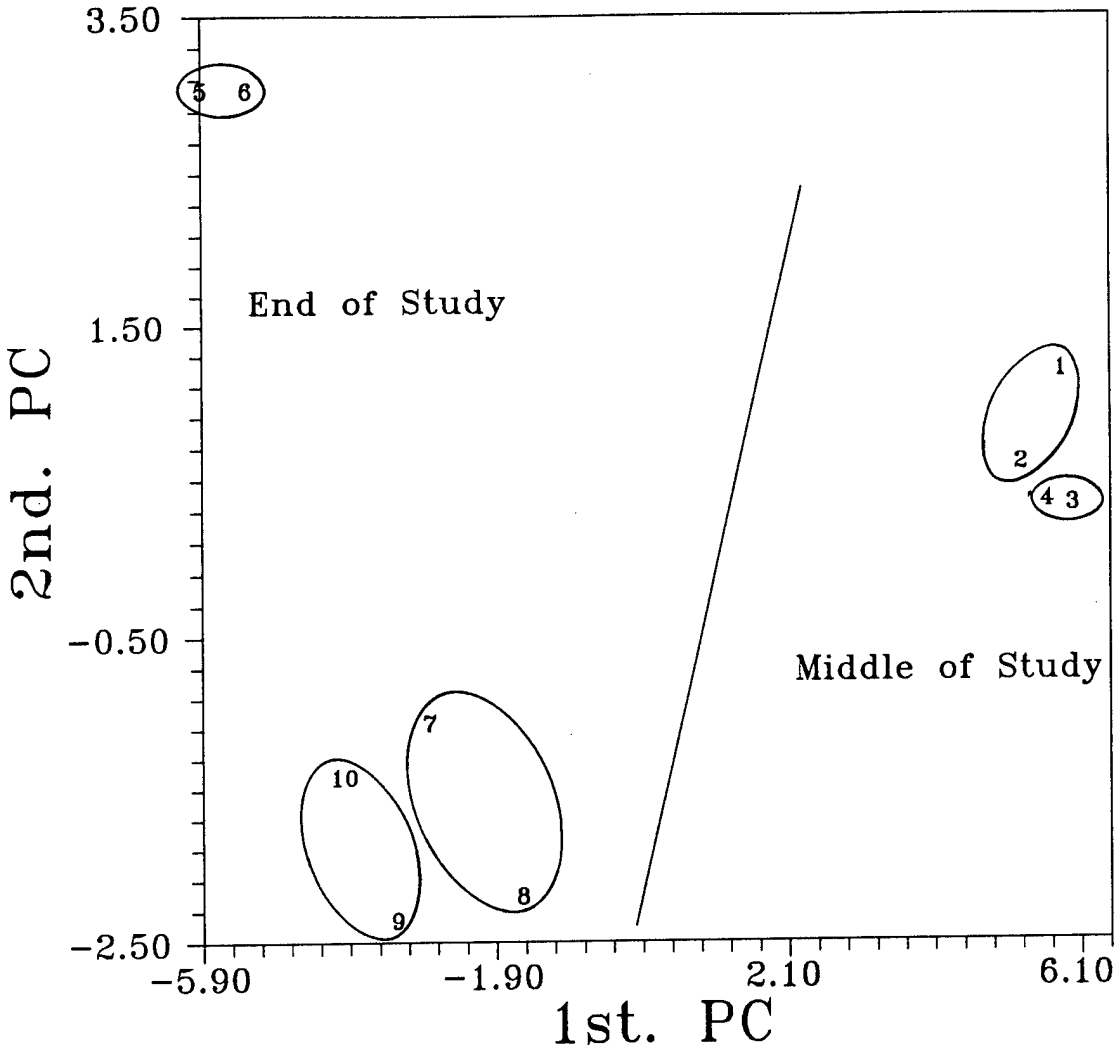


Figure 7. Principal Components Representation of the Pattern Space Defined by the 48 GC Peaks for the Diesel Fuel Data. The PC Map Accounts for 85 percent of the Total Cumulative Variance. Each Replicate Pair is Enclosed by an Elipse.

In the study it has been observed that the aging of the GC column introduces noise into the data that could influence the overall classification process. Other workers have recognized and commented on some of these difficulties in similar situations (23, 24). The confounding of the chemical information by this experimental artifact was investigated using principal component analysis. In Figure 7, the results of a principal components mapping experiment are shown for the Diesel fuel data. Water samples analyzed in the middle of the study are represented by the numbers 1, 2, 3, and 4 on the plot and cluster in one region of the principal components space

distant from the points corresponding to water samples analyzed near the end of the study (numbers 5 thru 10 in the plot). Since the variance in the data accounted for by the two largest principal components is 80 percent, the principal component map shown in Figure 7 is a reasonable representation of the distribution of the Diesel fuel data points in the higher dimensional measurement space and henceforth indicative of trends present in the data. The results of this principal component mapping experiment suggest that changes probably have occured in the GC column over time and that the variability introduced into the experiment by this factor is larger than replicate variability.

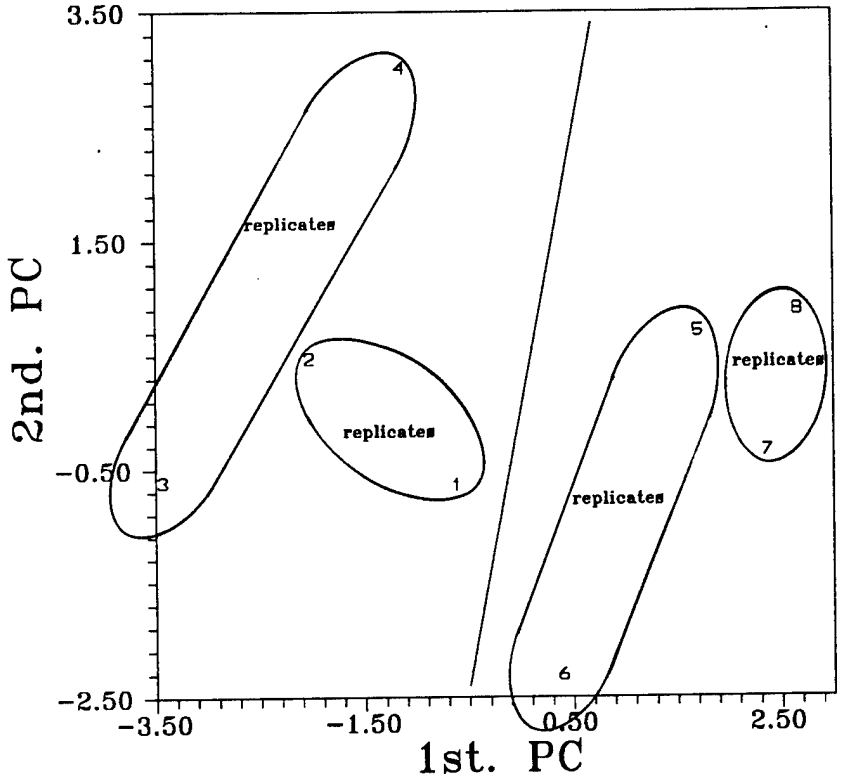


Figure 8. Principal Components Representation of the Pattern Space Defined by the 48 GC Peaks for JP-7. The PC Map Accounts for 90 percent of the Total Cumulative Variance. The Order in which the Samples were Run is 1, 2, 3, 4, 5, 6, 7, and 8. Each Replicate Pair is Enclosed by an Elipse.

Figure 8 shows a principal component map of the JP-7 data. Again, clustering of the data points on the basis of the order in which the samples were run is observed. However, the variability between JP-7 replicates is larger than Diesel fuel replicates. Since JP-7 has little aromatic content (and hence few water soluble components), the poorer precision observed for JP-7 is not surprising in view of the fact that we are probably outside the limits of quantitation for the method (the determination of the water soluble components of the fuel via extraction) with regard to this fuel.

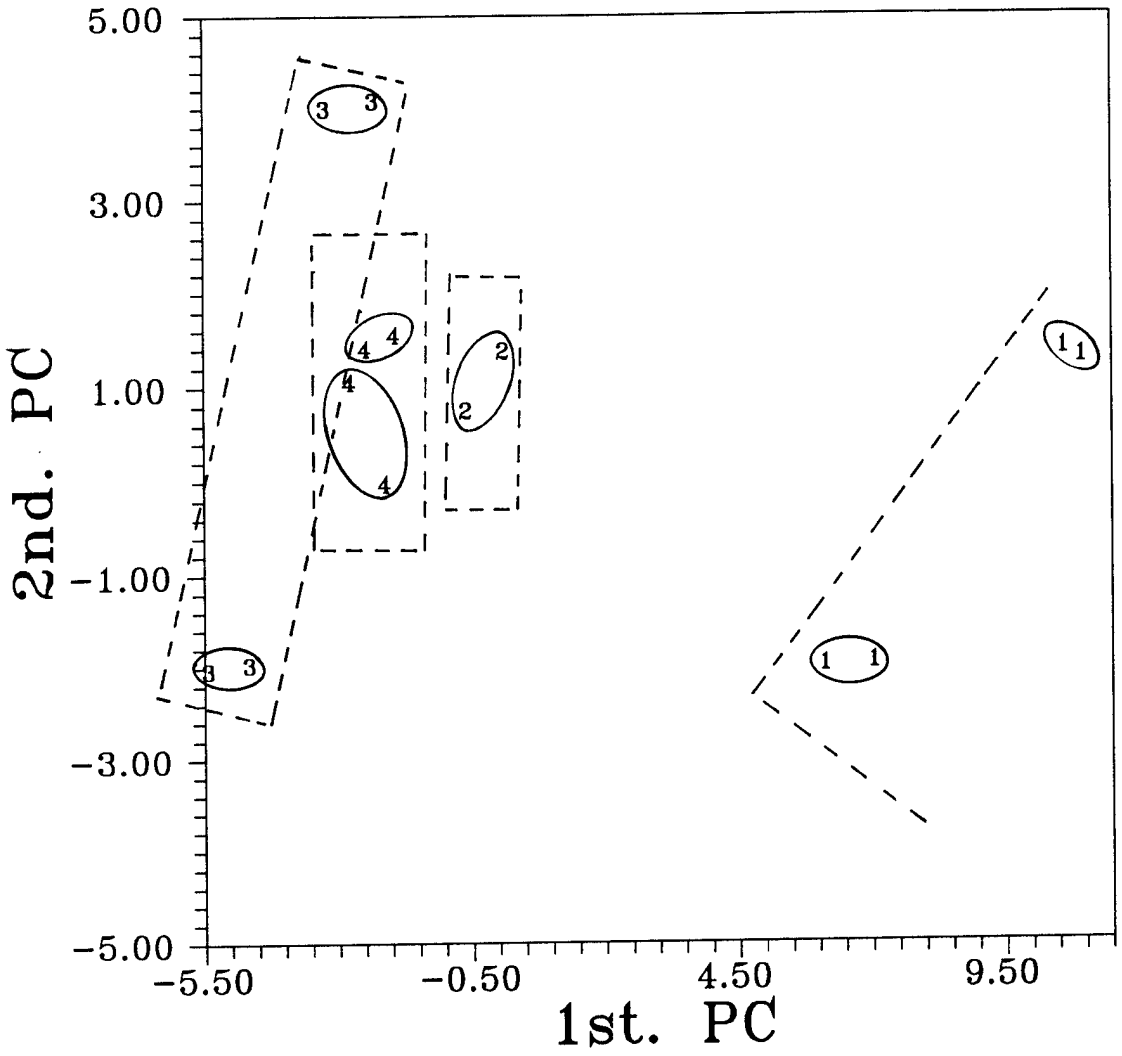


Figure 9. Principal Components Representation of the Pattern Space Defined by the 48 GC Peaks for JPTS. The PC Map Accounts for 88 percent of the Total Cumulative Variance. 1 = Samples Analyzed in the Early Stages of the Study; 2 = Middle of the Study; 3 = Nearer to the End of the Study; and 4 = Last Set of Samples Within This Group Analyzed by GS/MS. Each Replicate Pair is Enclosed by an Elipse.

Figure 9 shows a plot of the two largest principal components of the JPTS data. Again, the location of the data points in the principal component map clocks the order in which the samples were analyzed. However, principal component maps of Jet-A, JP-4 or AVGAS data do not show clustering of the data points on the basis of this unwanted experimental artifact (see Figures 10 thru 12).

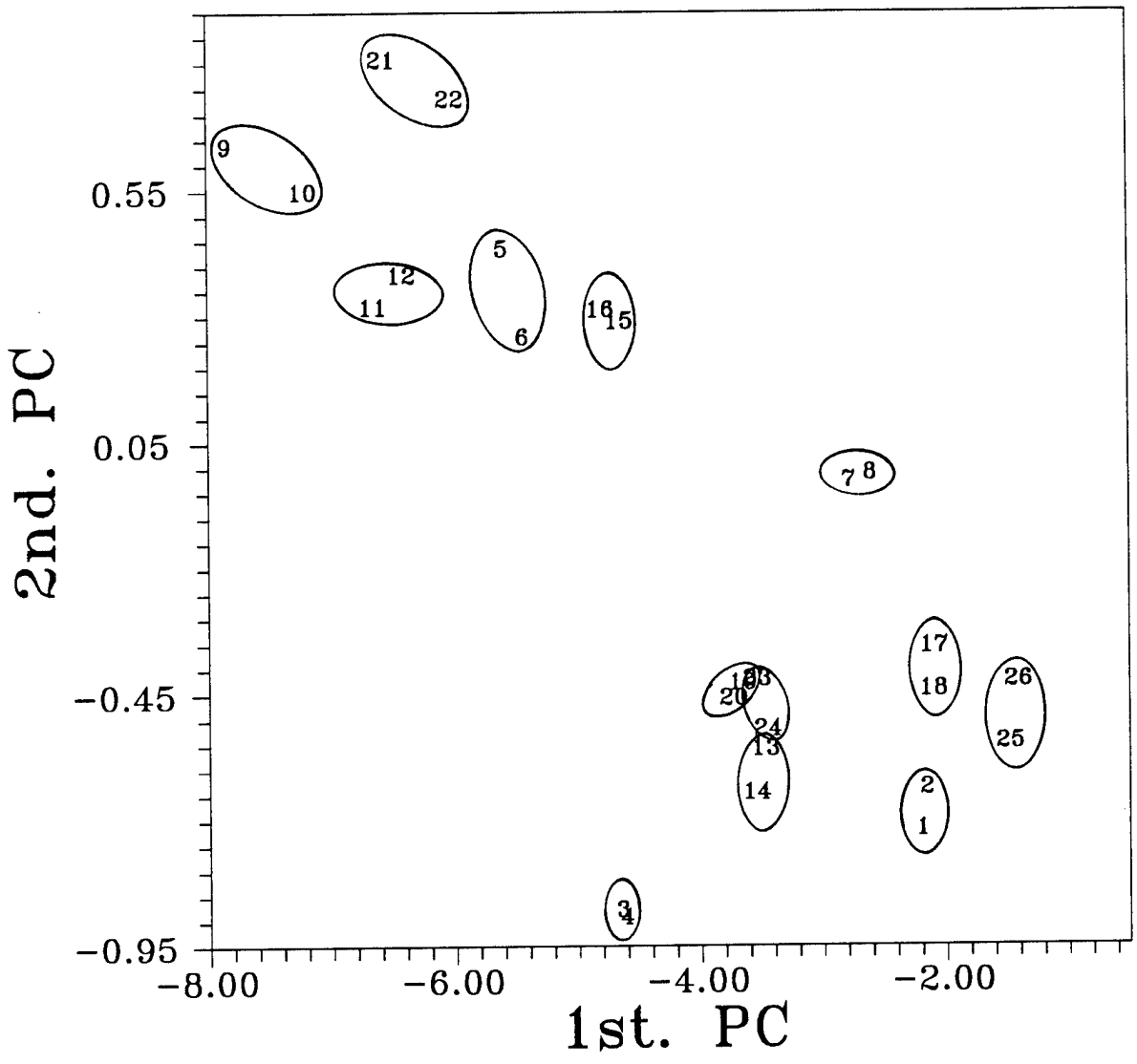


Figure 10. Principal Components Representation of the Pattern Space Defined by the 48 GC Peaks for Jet-A. The PC Map Accounts for 80 percent of the Total Cumulative Variance. The Order in Which the Samples were Run is 1, 2, 3, ... 26. Each Replicate Pair is Enclosed by an Elipse.

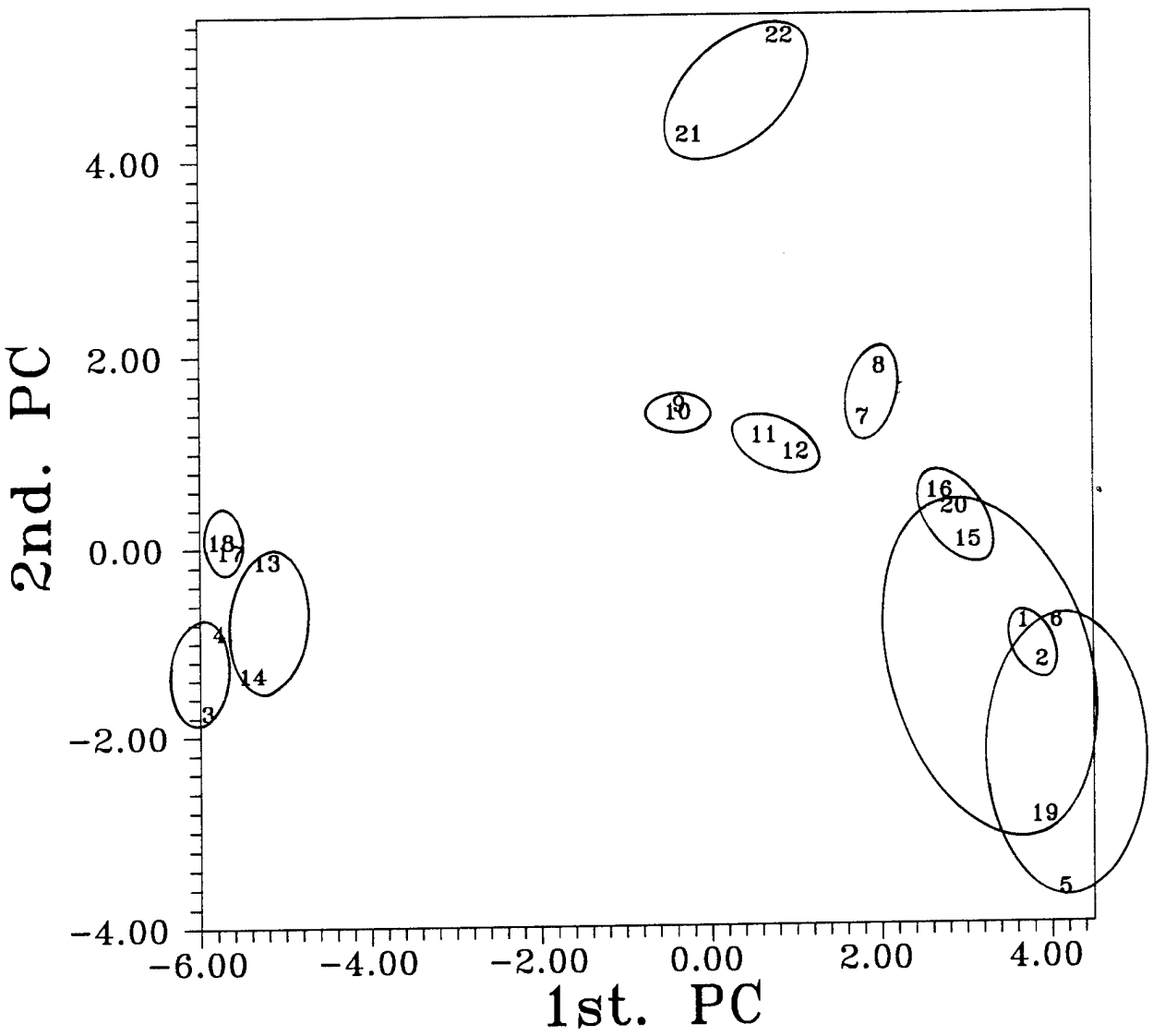


Figure 11. Principal Components Representation of the Pattern Space Defined by the 48 GC Peaks for JP-4. The PC Map Accounts for 68 percent of the Total Cumulative Variance. The Order in which the Samples were Run is 1, 2, 3, 4, 5, ... 26. Each Replicate Pair is Enclosed by an Elipse.

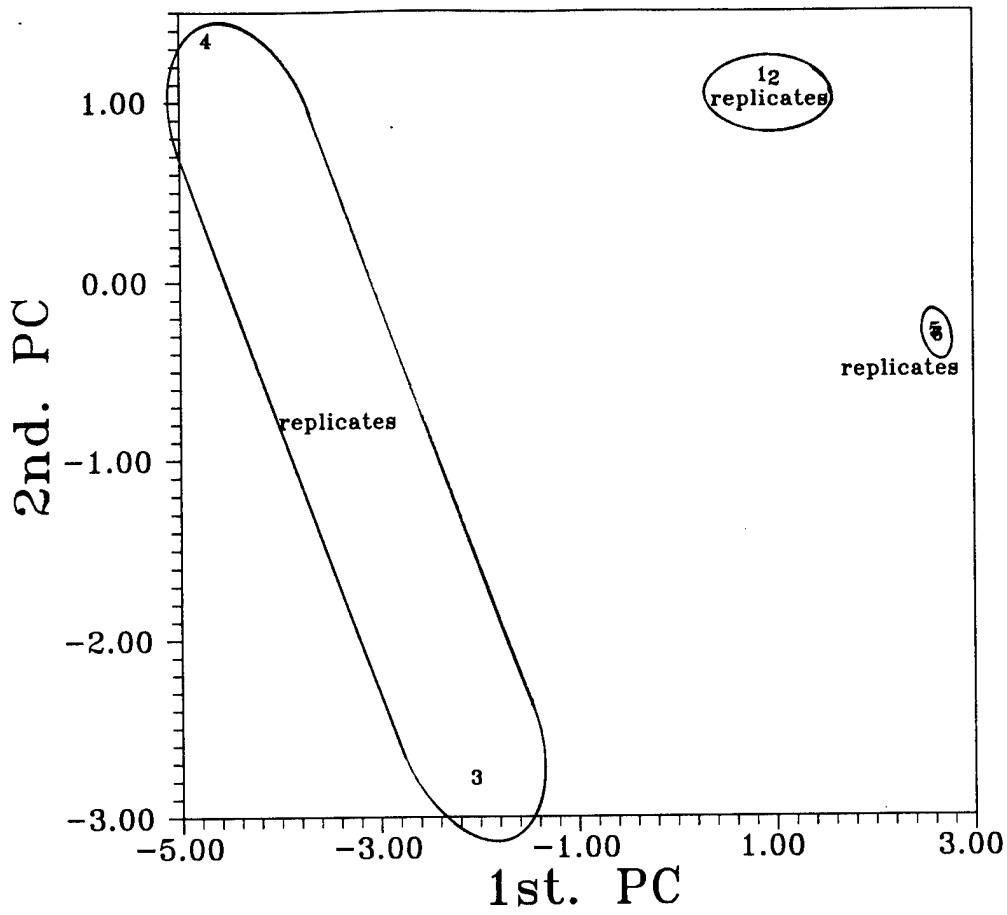


Figure 12. Principal Components Representation of the Pattern Space Defined by the 48 GC Peaks for AVGAS. The PC Map Accounts for 93 percent of the Total Cumulative Variance. The Order in Which the Samples were Run is 1, 2, 3, 4, 5, and 6. Each Replicate Pair is Enclosed by an Elipse.

Since the samples were not run in class-wise order during the study, it was possible to assess the effect of instrumental drift (i.e., aging of the GC column) on classification using supervised pattern recognition techniques, e.g., SIMCA. The SIMCA method is a subspace method based on a truncated principal components expansion of the data. Each class in the data set is represented by a principal component model. The complexity of each principal component model is determined directly from the data using the technique of cross-validation. Samples in the training set are fitted to each of the class models, and classifications are made according to the goodness of the fit. One of the advantages of SIMCA is that it can effectively use correlated variables. In fact, classification in SIMCA is based on differences in the correlation structure between the classes.

The test data consisted of 72 total ion chromatograms representing water samples that have been contaminated by JP-4, Jet-A, JPTS or Diesel fuel. The data were divided into four classes according to fuel type, and principal component models were developed for each class. (Water samples contaminated by either JP-7 or

AVGAS were omitted from this test data set since their chromatograms can be differentiated by visual observation from the other fuels.) Confidence envelopes around each class model were constructed for the purpose of containing the data points. The parameters of the confidence envelopes were derived from the principal components analysis. Goodness of fit statistics, e.g., the residual standard deviation, were then computed for the samples in the training set. An F-test was performed using the sample's residual standard deviation, together with the standard deviation of the class to determine whether or not a sample was inside or outside its assigned class and outside the other classes in the data.

TABLE 7. DESCRIPTORS USED FOR SIMCA PATTERN RECOGNITION

Peak Number	Compound
5	Benzene
8	Toluene
10	Ethylbenzene
11	p-Xylene
12	m-Xylene
14	o-Xylene
23	1, 2, 4 trimethylbenzene
29	1, 2, 3 trimethylbenzene
31	----------
32	----------
33	Indane
37	----------
40	----------
42	----------
43	tetralin
46	----------
47	methylnaphthalene

The modelling power (12) and discriminatory power (12) of each GC peak were also computed. The test data were then polished by deleting descriptors with low modelling power (less than 0.30) and low discriminatory power (less than 3.0) since these variables introduce unwanted noise in the class models and hence will contribute to a less efficient classification of the data. This feature selection process yielded 17 descriptors (see Table 7). The principal component models were then recomputed for the reduced test data, and classifications were made according to the goodness of fits using the standard deviation criterion. The results of this study are summarized in Table 8 and are in good agreement with a principal component map of the data (see Figure 13).

TABLE 8. SIMCA CLASSIFICATION RESULTS

Class	Number of PCs	NIC	*Right	Wrong	PCT Right
JP-4	2	22	22	0	100
Jet-A**	2	26	24	2	92
JPTS	2	14	14	0	100
Diesel	2	10	10	0	100
Total		72	70	2	97.2

* Of the 72 chromatograms, 62 were uniquely classified (11): JP-4 = 22, Jet-A = 16, JPTS = 14, and Diesel = 10.

** The two chromatograms misclassified were from the same sample.

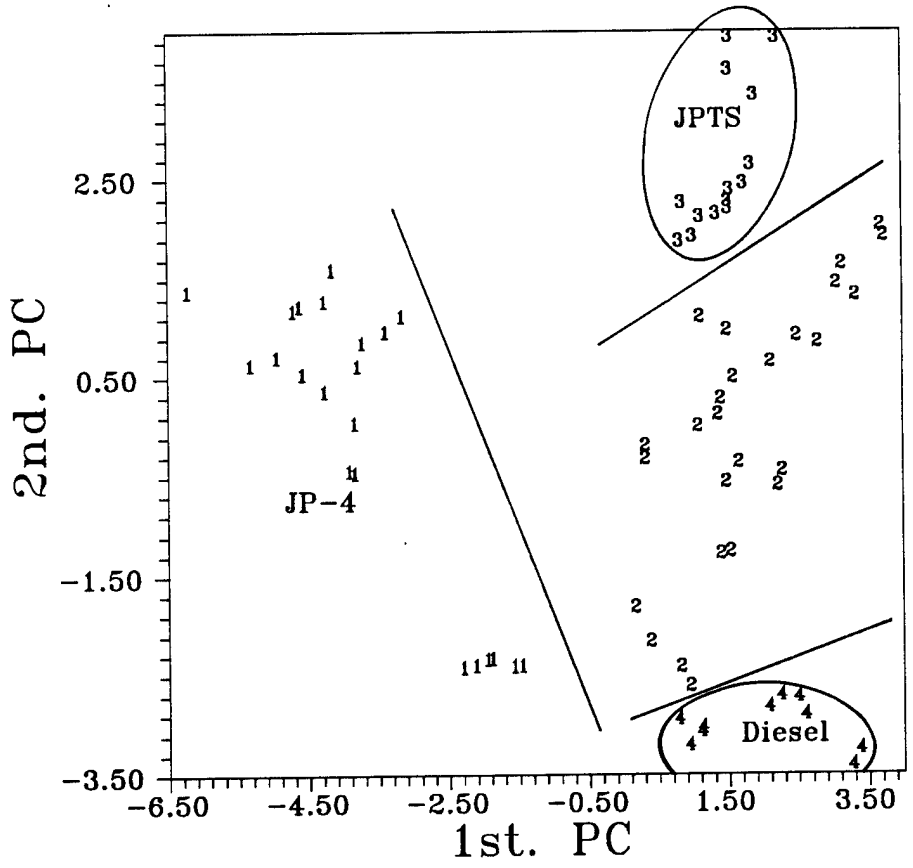


Figure 13. Principal Components Representation of the Pattern Space Defined by the 17 GC Peaks used for SIMCA Analysis. The PC Map Accounts for 68 percent of the Total Cumulative Variance. 1 = JP-4; 2 = Jet-A; 3 = JPTS; and 4 = Diesel.

Since a principal component is a linear combination of the original measurement variables, a plot of the principal component loadings, that is the coefficients of each of the GC peaks used for construction of the principal components, can also reveal information about trends present in the data. Figure 14 shows a plot of the loadings of the first two principal components of the reduced test data (i.e., the 17 descriptor data). The location of the GC peaks in the plot clock elution order suggesting confounding of the desired class information by an unwanted experimental artifact, column aging.

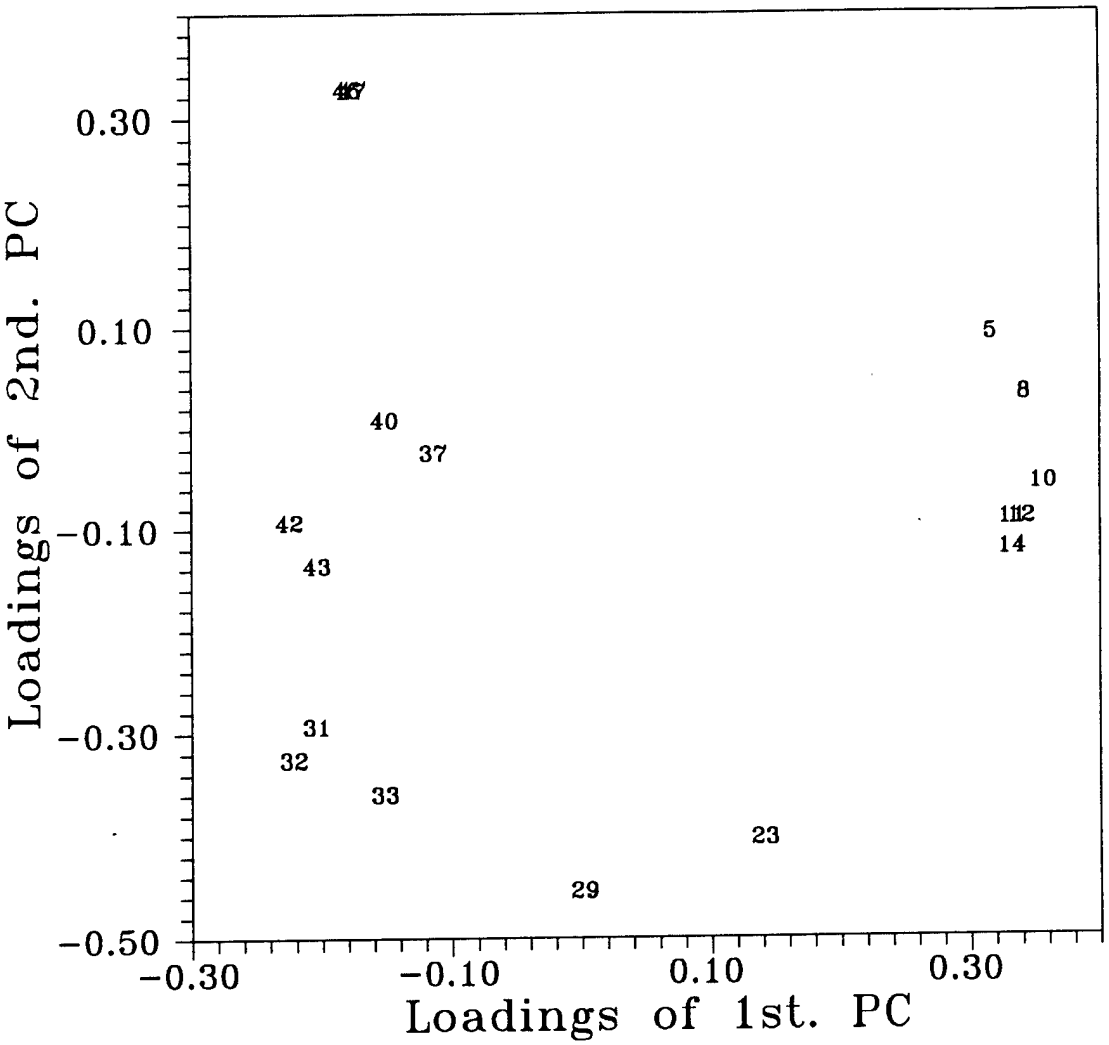


Figure 14. A Loading Plot of the Two Largest Principal Components for the Reduced Test Data.

The following experimental sequence was used to assess the contribution of this experimental artifact to the overall classification process. First, six sets of chromatograms were developed by random selection from the 72 chromatogram data set, where the training set contained 66 chromatograms and the prediction set contained the remaining 6 chromatograms. Any particular sample was only present in one of the six prediction sets generated. Principal component models were developed from the training set data and tested on the prediction sets. The average correct classification for the prediction sets was 94.5 percent, i.e., 68 out of 72 samples were correctly classified using the sd criterion. This same experiment was repeated again except that members of the prediction set included samples analyzed in the same time period. The average correct classification for the prediction set in this group of runs was 89 percent, i.e., 64 out of 72 samples were correctly classified using the sd criterion. Although the predictive ability of the principal component models was diminished when this confounding effect was taken into account, favorable classification results were still obtained.

To further test the predictive ability of these descriptors and the principal component models associated with them, a prediction set of 8 chromatograms was employed (see Table 6). The samples constituting the prediction set were analyzed on different days. A classification success-rate of 100 percent was achieved for the prediction set which suggests that the aging of the GC column is not as serious a problem as previously believed (23, 24). The studies presented for the 72-chromatogram data set also demonstate the feasibility of applying subspace methods to underdetermined data.

The feature selection process used in this study to develop the principal component models was not based on discriminatory power alone, e.g., differences in means between classes of samples. Selection of variables on the basis of discriminatory power alone, especially in the case of under-determined data sets (i.e., data sets in which the number of independent samples is smaller than the number of descriptors) would lead to a gross exaggeration of the differences between the classes and hence misleading results.

B. CONCLUSIONS

In this report, a basic methodology for analyzing complex chromatographic data sets was described. A chromatogram was represented as a point in a high-dimensional space. Pattern recognition methods were then used to investigate the properties of this vector space. The techniques found to be most useful in the studies reported here are nonparametric in nature. As such, they do not attempt to fit the data to an exact functional form; rather relationships are sought which provide definitions of similarity between diverse groups of data.

The methodology was successfully used to analyze three different data sets. In one study, a potential method for identifying neat jet fuels was developed using gas

chromatography and SIMCA pattern recognition. During the study, confounding relationships within the complex multivariate data were uncovered using FCV-false color data imaging. This was not unexpected since in any GC profile study, a complicating aspect of the classification of the samples will be the confounding of the desired group information by experimental variables or other systematic variations in the data. If the basis of classification for patterns in the training set is other than the desired group difference, unfavorable classification results for the prediction set are obtained despite a linearly separable training set. The existence of these types of complicating relationships will be an inherent part of fingerprint-type data.

In a second study, the FCV clustering algorithm was used to partition fuel samples into different classes on the basis of fuel type. The test data consisted of total ion chromatograms of neat jet fuels and binary mixtures of the fuels. Because the components of many of the binary fuel mixtures were correctly identified, we conclude that the FCV clustering algorithm can be used for linear mixture analysis. The capability to identify samples which are mixtures of different types of fuels is especially critical in view of the economic liability incurred by the party or parties responsible for the fuel spill.

In a third study, gas chromatography and pattern recognition techniques were used to develop a potential method to identify sources of pollutants dissolved in water. The water samples were prepared by placing a neat jet fuel in contact with distilled water in a vessel designed by McIntyre. Total ion chromatograms of the dissolved hydrocarbons were analyzed by pattern recognition techniques and were found to be characteristic of fuel type. The results of this study suggest that gas chromatograms of dissolved hydrocarbons can be used to identify the original fuel. Hence, it should be possible to monitor a suspected well and determine the type of fuel responsible for the contamination through pattern recognition analysis of the dissolved hydrocarbon profile.

C. RECOMMENDATIONS

Pattern recognition is performed through application of four different operations: transduction, preprocessing, feature selection, and classification. The work described in this report was directed towards the development of better classification methods for fingerprint data. However, the confounding of the desired group information by experimental artifacts in fingerprint data is a serious problem that needs to be addressed. Hence, research should be directed towards the development of feature selection methods that minimize these unwanted relationships within the data. Since statistics cannot offer an eloquent solution to this problem save experimental design, brute force calculations based on the K-nearest neighbor method are justified. Such a calculation would identify a set of features that maximize differences between the classes while ensuring that nearest neighbors in the feature space are not samples that were analyzed on the same day. The availability of supercomputing power via workstations, e.g., IBM RISC 6000 workstations, makes this approach attractive and practical.

REFERENCES

1. Jurs, P.C., B.K. Lavine, and T.R. Stouch, "Pattern Recognition Studies of Complex Chromatographic Data Sets," NBS Journal of Research, December 1985, pp. 543-549.

2. Belcher, A.M., A.B. Smith, P.C. Jurs, B.K. Lavine, and G. Epple, "Analysis of Chemical Signals in a Primate Species (Saguinus fusciocollis): Use of Behavior, Chemical, and Pattern Recognition Methods," Journal of Chemical Ecology, May 1986, pp. 513-522.

3. Lavine, B.K. and D. Carlson, "European Bee or Africanized Bee? Species Identification Through Chemical Analysis," Analytical Chemistry, March 1987, 59, pp. 468A- 472A.

4. Dunn, W.J., D.L. Stalling, T.R. Schwartz, J.W. Hogan, and J.D. Petty, "Pattern Recognition for Classification and Determination of Polychlorinated Biphenyls in Environmental Samples," Analytical Chemistry, vol. 56, September 1984, pp. 1308-1315.

5. Nilson, N.J. Learning Machines, McGraw-Hill, New York, 1965.

6. Lachenbruch, P.A. Discriminant Analysis, Hafner Press, New York, 1975.

7. Dunn, W.J., S. Wold, and Y.C. Martin, "Structure-Activity Study of Beta-Adrenegic Agents Using the SIMCA Method of Pattern Recognition," Journal of Medicinal Chemistry, vol. 21, May 1978, pp. 922-932.

8. Lavine, B.K., P.C. Jurs, D.R. Henry, "Chance Classifications by Nonparametric Linear Discriminant Functions," Journal of Chemometrics, vol. 2, January 1988, pp. 1-10.

9. Lavine, B.K., and D.R. Henry, "Monte Carlo Studies of Nonparametric Linear Discriminant Functions," Journal of Chemometrics, vol. 2, January 1988, pp. 85-90.

10. Bezdek, J.C., C.R. Coray, R.W. Gunderson, and J.D. Watson, "Detection and Characterization of Cluster Substructure I. Linear Structure: Fuzzy c-Lines," SIAM Journal of Applied Mathematics, vol. 40, April 1981, pp. 339-357.

11. Bezdek, J.C., C.R. Coray, R.W. Gunderson, and J.D. Watson, "Detection and Characterization of Cluster Substructure II," SIAM Journal of Applied Mathematics, vol. 40, April 1981, pp. 358-372.

12. Wold, S. and M. Sjostrom, "SIMCA: A Method for Analyzing Chemical Data in Terms of Similarity and Analogy," B.R. Kowalski, Editor, Chemometrics: Theory and Application, vol. 52, pp. 243-280, American Chemical Society, Washington DC, 1977.

13. Jolliffe, I. P., Principal Component Analysis, Springer-Verlag Press, New York, 1986.

14. Ekftrom, M.P. Digital Imaging Techniques, Academic Press, Orlando Florida, 1984.

15. Leu, C.H., and D.S. Bowies, Editors, Procedures of the Vancouver Symposium, IAHS Publication No. 166, pp. 56-70, August 1987.

16. Gunderson, R.W., Editor, 5th Annual Nordic Conference on Applied Statistics, pp. 65-85, Stockkand Foriag Publishers, Stravanger Norway, 1985.

17. Lavine, B.K., L. Morel, R.K. Vander Meer, R.W. Gunderson, J.H. Han, A. Bonanno, and A. Stine, "Pattern Recognition Studies in Chemical Communication:Nestmate Recognition in *Camponotus floridanus*," Chemometrics and Intelligent Laboratory Instrumentation, vol. 9, September 1990, pp. 107-114.

18. Kreyszig, E. Advanced Engineering Mathematics, 4th Ed., p. 54, John Wiley & Sons, New York, 1979.

19. Mayfield, H.T. and M.V. Henley, "Classification of Jet Fuels using High Resolution Gas Chromatography and Pattern Recognition," Hall, J.R and G.D. Glysson, Editors, Monitoring Water in the 1990's: Meeting New Challenges, ASTM STP 1102, pp. 578-597, American Society for Testing and Materials, Philadelphia PA, 1991.

20. Mayfield, H.T., and W. Bertsch, "Set-Up: A Program for Peak Matching," Journal of Computer Applications in the Laboratory, vol. 1, January 1983, pp. 130-137.

21. Stone, M.,"Cross-Validitory Choice and Assessment of Statistical Predictions," Journal of the Royal Statistical Society, vol. 36, February 1974, pp. 111-121.

22. Burris, D.R. and W.G. MacIntyre, "Water Solubility Behavior of Binary Hydrocarbon Mixtures," Environmental Toxicology and Chemistry, vol. 4, no. 3, 1985, pp. 371-377.

23. Pino, J.A., J.E. Murray, P.C. Jurs, B.K. Lavine, and A.M. Harper, "Application of Pyrolysis/Gas Chromatography/Pattern Recognition to the Detection of Cystic Fibrosis Heterozygotes," Analytical Chemistry, vol. 57, 1985, pp. 295-302.

24. Blomquist, G., E. Johnson, B. Soderstrom, and S. Wold, "Classification of Fungi by Means of Pyrolysis-Gas Chromatography-Pattern Recognition," Journal of Chromatography, vol. 173, 1979, pp. 19-32.

BIBLIOGRAPHY

Adobe Systems Incorporated, PostScript Language Reference Manual, Addison Wesley Publishing Company Inc., California, 1986.

Asente, P.J. and R.R. Swick, with J. McCormack, X Window System Toolkit: The Complete Programmers Guide and Specification X Version 11 Release 4, Digital Press, Maynard Massachusetts, 1990.

Digital Equipment Corp, DEC C Language Reference Manual, Version 1.0, Digital Equipment Corp, Maynard Massachusetts, 1991.

Digital Equipment Corp, DEC C User's Guide for ULTRIX Systems, Version 1.0, Digital Equipment Corp, Maynard Massachusetts, 1991.

Digital Equipment Corp, DEC FORTRAN Language Reference Manual, Version 3.0, Digital Equipment Corp, Maynard Massachusetts, 1991.

Digital Equipment Corp, DEC FORTRAN for ULTRIX RISC Systems User Manual,Version 3.0, Digital Equipment Corp, Maynard Massachusetts, 1991.

Digital Equipment Corporation, XUI Style Guide, Digital Equipment Corporation, Maynard Massachusetts, 1988.

Digital Equipment Corporation, XUI Programming Overview, Digital Equipment Corporation, Maynard Massachusetts, 1990.

Digital Equipment Corporation, Guide to the XUI User Interface Language Compiler, Digital Equipment Corporation, Maynard Massachusetts, 1990.

Digital Equipment Corporation, Guide to the XUI Toolkit: C Language Binding, Digital Equipment Corporation, Maynard Massachusetts, 1990.

Digital Equipment Corporation, Guide to the XUI Toolkit Widgets: C Language Binding, Digital Equipment Corporation, Maynard Massachusetts, 1990.

Digital Equipment Corporation, Guide to the XUI Toolkit Intrinsics: C Language Binding, Digital Equipment Corporation, Maynard Massachusetts, 1990.

DuToit, S.H.C., A.G.W. Steyn, and R.H. Stumpf, Graphical Exploratory Data Analysis, Springer-Verlag, New York, 1986.

Foley, J.D., A. Van Dam, S.K. Feiner, and J.F. Hughes, Computer Graphics; Principals and Practice, 2nd Edition, Addision-Wesley Publishers Inc., Massachusetts, 1990.

Heller, Dan, The Definitive Guides to the X Window System: Motif Programming Manual for OSF/Motif Version 1.1, Motif Edition, Volume 6, O'Reilly and Associates Inc., California, 1991.

Newman, W. M., and R.G. Sproull, Principles of Interactive Computer Graphics, 2nd Edition, McGraw Hill, New York, 1989.

Nye, Adrian, The Definitive Guides to the X Window System: Xlib Programming Manual for Version 11, Volume One, O'Reilly and Associates, Inc., California, 1990.

Nye, Adrian, Editor, The Definitive Guides to the X Window System: Xlib Reference Manual for Version 11, Volume Two, O'Reilly and Associates, Inc., California, 1990.

Open Software Foundation, OSF/Motif Programmer's Reference, Revision 1.1, Open Software Foundation, Prentice Hall Inc., New Jersey, 1991.

Open Software Foundation, OSF/Motif Programmer's Guide, Revision 1.1, Open Software Foundation, Prentice Hall Inc., New Jersey, 1991.

Open Software Foundation, OSF/Motif Style Guide, Revision 1.1, Open Software Foundation, Prentice Hall Inc., New Jersey, 1991.

Open Software Foundation, OSF/Motif Style Guide, Revision 1.1, Open Software Foundation, Prentice Hall Inc., New Jersey, 1991.

Open Software Foundation, OSF/Motif User's Guide, Revision 1.1, Open Software Foundation, Prentice Hall Inc., New Jersey, 1991.

Preparata, F.P., and M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, New York, 1985.

Sheifler, R.W. and J. Gettys, X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD, X Version 11, Release 4, Digital Press, Massachusetts, 1990.

Watt, A.H., Fundamentals of Three-Dimensional Computer Graphics, Addison-Wesley Publishers Ltd., Great Britain, 1990.

Young, D.A., The X Window System: Programming and Applications with Xt, OSF/Motif Edition, Prentice Hall Inc., New Jersey, 1990.

(The reverse of this page is blank.)

APPENDIX A:

XFCV Data Analysis and Visualization System

Users Manual

(The reverse of this page is blank.)

XFCV Data Analysis and Visualization System

Users Manual

April, 1992

Revision/Update Information: V2.1 (supersedes V2.0 documentation)

Platforms supported:

VAX/VMS 5.3-1 or greater w/ DECwindows V2 and DEC Motif Developers Kit V1.1.1 or greater

ULTRIX/RISC 4.x w/ DECwindows V2 and DEC Motif Developers Kit V1.1.1 or greater

SunOS 4.1.1 (Solaris 1.0) w/ OpenWindows V2.0/V3.0 and DECwindows Motif for the SUN SPARCstation V1.0 or greater

April 1992

Contents

Figures

Preface

The *XFCV User's Manual* provides complete details on the use of the **XFCV False Color Data Imaging and Visualization System.**

Intended Audience

This manual is intended for all users of the **XFCV False Color Data Imaging and Visualization System.**

Document Structure

This manual is organized into two major parts. Chapter 1 and 2 provide the details of the makeup of the system and discuss the available options. Chapter 3 provides a sample use of the system, using the *Iris*[1] data set. The manual makes liberal use of illustrations to show the reader what the system looks like and how it works.

---

1 A set of iris measurement usually attributed to Fisher has been widely used as a *defacto* standard test for many data reduction techniques. The iris data collected by Anderson [1935] and first analyzed by Fisher [1936] consists of four measurements on 150 flowers. The septal and petal lengths and widths were detemined for 50 flowers of each of three varieties of iris.

Associated Documents

You may find additional useful information on the concepts and use of XFCV in the following documents:

Adobe Systems Incorporated, PostScript Language Reference Manual, Addison Wesley Publishing Company Inc., California, 1986.

DuToit, S.H.C., Steyn, A.G.W., and Stumpf, R.H., Graphical Exploratory Data Analysis, Springer-Verlag, New York, 1986.

Open Software Foundation, OSF/Motif Style Guide, Revision 1.1, Open Software Foundation, Prentice Hall Inc., New Jersey, 1991.

Open Software Foundation, OSF/Motif User's Guide, Revision 1.1, Open Software Foundation, Prentice Hall Inc., New Jersey, 1991.

Preparata, Franco P., and Shamos, Michael I., Computational Geometry: An Introduction, Springer-Verlag, New York, 1985.

Watt, Alan H., Fundamentals of Three-Dimensional Computer Graphics, Addison-Wesley Publishers Ltd., Great Britain, 1990.

Conventions

Convention	Meaning
Ctrl/x	A sequence such as Ctrl/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 x	A sequence such as PF1 x indicates that you must first press and release the key labeled PF1, then press and release another key or pointing device button.
MB1, MB2, MB3	Indicates that you press a mouse button (MB). The ordering of the buttons on the mouse depends on the orientation of the mouse: for a right-handed mouse, MB1=left-most button, MB3=right-most button; for a left-handed mouse, MB1=right-most button, MB3=left-most button.
RET	A key is shown enclosed to indicate that you press a key on the keyboard.
...	In examples, a horizontal ellipsis indicates one of the following possibilities: - Additional optional arguments in a statement have been omitted. - The preceeding item or items can be repeated one or more times. - Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed or there is not sufficient room to show all items.
()	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[]	In format descriptions, brackets indicate that whatever is enclosed is optional; you can select none, one, or all of the choices.
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.

boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason.
italic text	Italic text represents information that you can vary in system messages (for example, Internal error *number*). Italics can also specify references to other sources of information (e.g., "refer to the *XFCV Installation Guide* ..."). Italic text is also used to show user input, in contrast to system output, in examples showing a dialog with the system.
UPPERCASE TEXT	Uppercase letters indicate that you must enter a command (for example, enter RUN XFCV). Uppercase letters can also indicate the name of a command, the name of a file, etc.
numbers	Unless otherwise noted, all numbers in the text are assumed to be decimal. Nondecimal radixes - binary, octal, or hexadecimal - are explicitly indicated.

Glossary

The following terms, abbreviations, and symbols are used in this manual:

DEC	Digital Equipment Corporation, major manufacturer of a wide range of computer systems.
DECwindows	DEC's enhanced X Window windowing system for both their VMS and Ultrix based systems.
GC	Gas Chromatograph. This is an instrument used to analyze samples and determine their makeup. This is often the source of raw data for the XFCV system (in terms of sample measurements).
GUI	Graphical User Interface. Layered on top of a windowing system (such as X), the GUI defines a *look and feel* for applications using the windowing system.
HPLC	High Performance Liquid Chromatography. This is a method that, like the GC, is used to gain information about the chemical makeup of a compound.
Motif	Short for OSF/Motif, this is the GUI style designed by the Open Software Foundation.
OPEN LOOK	Sun/AT&T graphical user interface definition.
OpenWindows	Sun's implementation of the OPEN LOOK definition.
Solaris	Sun's operating system/user environment on their SPARC series of workstations.
Sun	Sun Microsystems Inc, major manufacturer of SPARC based workstations and systems.
Ultrix	DEC's enhanced Unix operating system for their VAX and RISC based computers.
Unix	Developed originally by AT&T, this is the *defacto* standard operating system on workstations.

VAX/VMS	DEC's operating system for their VAX series of computers (including mainframes, minicomputers, and workstations).
X	X Window System from MIT. This is the windowing system XFCV uses on all workstation platforms (VAX/VMS, Unix)

Chapter 1

Introduction: XFCV Concepts and Definitions

If a scientist is seeking the relationship between a physical property and the concentration of a species in a sample, a simple plot of property vs. concentration usually leads to the correct functional relationship. The modern scientist, however, is becoming increasingly involved with experimental design and measurement systems (e.g., GC/MS or HPLC). Therefore, data analysis often means finding relationships, not only between the property of interest and the composition of the sample, but also developing relationships between sets of chemical measurements (e.g., areas of selected GC peaks) and the class assignment of the sample (e.g., origin of an environmental pollutant).

The **XFCV False Color Data Imaging and Visualization System** provides the user with a concise tool for the analysis and visualization of multivariate data sets, such as those obtained in

1.1 How to Use This Manual

The *XFCV User's Manual* is organized into two major parts. Chapters 1 and 2 provide background on the XFCV system and detail the user interface. Chapter 3 provides an example of using the system to analyze a sample data set. A large number of screen images are provided in order to show the reader what the system looks like. Note that the images in Chapters 1 and 2 are not shown in color; screen images of the sample data set in Chapter 3 are shown in full color. The actual color of your screen and data may vary, depending on system configuration. In addition, the appearance of text (fonts) in the system will vary from workstation to workstation, depending on availability fonts.

1.2 Statistical Methods Employed by XFCV

There are three primary statistical methods employed by XFCV to perform the analysis and visualization of multivariate data: principal component analysis, fuzzy c-variety (FCV) clustering, and false color data imaging.

1.2.1 Principal Component Analysis

Graphical methods are often used by physical scientists to study data. If there are only two or three measurements per sample, the data can be displayed as a graph for direct viewing. In otherwords, the data can be displayed as points

in a two- or three-dimensional measurement space. The coordinate axes of the space are defined by the measurement variables. By examining the graph, a scientist can search for similarities and dissimilarities among the samples, find natural clusters, and even gain information about the overall structure of the data set. If there are n-measurements per sample (n > 3), a two- or three-dimensional representation of the measurement space is needed in order to visualize the relative position of the data points in n-space. This representation must accurately reflect the high dimensional structure of n-space. One such approach is to use a mapping and display technique called principal component analysis.

Principal component analysis[12] is a method for transforming the original measurement variables into new, uncorrelated variables called principal components. Each principal component (PC) is a linear combination of the original measurement variables. Using this procedure is analogous to finding a set of orthogonal axes which represent the directions of greatest variance in the data. (The variance is defined as the degree to which the data points are spread apart in the n-dimensional measurement space.) If a data set has a large number of interrelated variables, then principal component analysis is a powerful method for analyzing the structure of that data and reducing the dimensionality of the pattern vectors.

The procedure for implementing principal component analysis is as follows. First, the covariance matrix of the data set is computed:

$$S=\frac{1}{N-1}X^{T}X \qquad (2)$$

where X is the data matrix, X^T is the transpose of the data matrix, and N is the dimensionality of the data. Next, an eigenanalysis is performed on the covariance matrix. The eigenvector corresponding to the largest eigenvalue represents the direction of greatest variance in the data; each successive eigenvector represents the direction of maximum residual variance. The eigenvectors (i.e. principal components) are then arranged in order of decreasing variance - the first eigenvector is the most informative and the last eigenvector is the least informative. The two largest eigenvectors or principal components are retained; the values of the two largest principal components are computed for each data point. Finally, the points are projected onto a plane defined by the two largest principal components. The amount of information in the principal component plot relative to the original measurement variables is expressed as

$$I = \frac{\lambda_1 + \lambda_2}{\sum_{K=1}^{N} \lambda_K} \tag{3}$$

where I is the fraction of the total cumulative variance, λ_K is the eigenvalue of the kth eigenvector, and N is the number of descriptors in the data matrix.

1.2.2 FCV Clustering Algorithm

The FCV clustering algorithms were first published by Bezdek in 1981. Since that time, they have been successfully used to analyze satellite and star-mapping data[13,14]. For profile analysis the method was first used to assess the role of cuticular hydrocarbons in nestmate recognition for social hymenoptera[15] and proved to be a useful tool. The FCV algorithm can be used to search for trends present in a data set and can cluster the data points according to these trends. The algorithm is most useful when the data structure is not known, as is sometimes the case in studies concerned with source apportionment of environmental pollutants.

An interesting feature of the FCV clustering algorithm is that each data vector in the training set is assumed to contribute to the modelling of each of the classes within the data. The actual algorithm consists of solving simultaneously the following set of equations using a *Picard iteration*

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{D_{ik}}{D_{jk}}\right)^{1/(m-1)}} \tag{4}$$

$$v_i = \frac{\sum_{k=1}^{n} (\mu_{ik})^m x_k}{\sum_{k=1}^{n} (\mu_{ik})^m} \tag{5}$$

$$S_i = \sum_{k=1}^{n} (\mu_{ik})^m (x_k - v_i)(x_k - v_i)^T \tag{6}$$

$$D_{ik} = (|x_i - v_i|^2 - \sum_{j=1}^{r} \langle x_k - v_i, d_{ij} \rangle^2)^{\frac{1}{2}} \tag{7}$$

The membership value of sample k with respect to cluster i ($i = 1, 2, 3, ...$) is μ_{ik}, and these values are subject to the conditions $0 < \mu_{ik} < 1$ and $\Sigma \mu_{ik} = 1$. D_{ik} is the distance of sample k from cluster center i, v_i is the center of cluster i, x_k is the sample data vector, d_{ij} is a unit eigenvector corresponding to the jth largest eigenvalue of the (fuzzy) within-cluster scatter matrix, S_i, and m is a fixed weighting exponent. To obtain an approximate solution to this set of four equations, the user must supply a starting class membership matrix. The cluster centers, the within-cluster scatter matrix for each cluster, and the distance of each sample from each cluster are computed in rapid succession. New membership values are then computed for the samples in the final step of the first iteration. The algorithm continues by using these new membership values as the starting matrix for a second iteration through the same set of equations. This process is allowed to continue until convergence is achieved. The number of iterations required to achieve convergence depends upon the minimum prespecified change criterion for the class membership values.

The value of m is usually set at 2, but by increasing m, less weight is attached to the importance of the samples with smaller membership values. The higher the value of m, the fuzzier the algorithm becomes in the sense that points whose membership values are uniformly low through the iterative procedure tend to become increasingly ignored in determining the membership functions and the defining linear varieties. It is this feature of the FCV algorithm that is particularly appealing when one suspects that the data may not exist in compact well separated clusters. The ability to "tune out" noise in the data by adjusting m can be of great value in obtaining favorable and meaningful clustering results for this type of data.

When investigating the data with the FCV clustering algorithm, one may choose to search for round clusters in the data by specifying $r = 0$ or find the best fit of the data to linear clusters by specifying $r = 1$, or in general attempt to fit the data to other geometric shapes by setting $r \geq 2$. This feature of the FCV algorithm allows the investigator to compare the fit of the data to a number of geometrically distinct cluster shapes and to select the fit which appears to best represent the actual structure of the data. Clearly, this feature results in a very definite advantage if one is interested in a careful analysis of the data.

1.2.3 FCV-False Color Data Imaging

In conjuction with the FCV clustering algorithm, which provides the class/cluster membership information for samples, one additional technique is used during the visualization process. The technique of *false color data imaging* (16) has long been used to analyze multispectral data measured by satellite electromagnetic scanner systems. In a false-color data imaging experiment, each data vector is projected onto a coordinate system defined by the three largest principal components. Each of the three principal axes is assigned a primary color, e.g., red, green, or blue. A given data point can then be assigned a color which is a combination of the three primary colors. The intensity of each primary color is inversely scaled according to the distance of the data point from the particular color axis in question. Thus, data projected onto a line equidistant from all three coordinate axes would be assigned a color composed of equal intensities of the three primary colors, i.e., some shade of gray.

The first step in an FCV-false color data imaging experiment is to project the original high-dimensional data onto a suitable three-dimensional subspace defined by the three largest principal components of the data. Since this is easily accomplished with microcomputer graphics even for large data sets, it is a reasonable first step in any data analysis. However, there is a problem associated with principal component maps. The spatial relationships between individual data points, and between groups of data points in the original measurement space are often distorted in the projective process. It is at this point where this method departs substantially from other graphical display techniques. By taking advantage of the membership coefficients generated by the FCV clustering algorithm, much of the spatial information lost during projection is restored through the use of color, a crucially important information dimension. Using the FCV clustering algorithm, the data is fitted to a user-specified number of linear disjoint principal component models. Each model, which represents a different cluster of data points, is assigned a different color: red, green, blue, etc. A given data point is displayed as a combination of these colors, with the amount of any one color determined by the sample's membership value for that particular class. Interpretation of the resulting color images provide valuable insight into the data structure. For example, two projected data points do not lie close to one another in the high-dimensional space (and hence are not similar) unless they are displayed in nearly the same color. A single cluster of data points in the three-dimensional principal component space, that appears to be made up of two distinct primary colors, will be interpreted as two distinct clusters whose true multidimensional separation has been lost through projection. A solid

group of data appearing in a non-primary color suggests the presence of an unsuspected class, and so forth.

1.2 Architecture of XFCV

XFCV is an integrated data analysis and visualization system designed for the investigation of of properties of multivariate (high-dimensional) data sets. XFCV is comprised of a small number of distinct modules to accomplish data editing & transformations, data analysis, and data visualization. The following shows the step-by-step process of using the XFCV components:

Data Editing/Creation <--> Transformations (scaling) --> Analysis <--> Visualization

1.3 Data Editing/Creation

XFCV provides you with various options for data editing/creation. The system utilizes a spreadsheet application to allow you to manage data. In V2.x, XFCV on workstation platforms incorporates a modified version of the public-domain spreadsheet *sc*. You may edit the raw data with anything you wish (text editor, another spreadsheet, etc). In general, data is arranged as rows of data vectors, with each vector containing the descriptors for that sample. Each data vector is proceeded by some identifying integer, to allow the sample to be uniquely identified among all other samples. Each data vector is assumed to comprise of floating point descriptor values; if a descriptor is specified as an integer (e.g, 1, 2, etc), it will be converted to a floating point value by the spreadsheet. Also, the spreadsheet will impose the same precision on all descriptors and all data samples (thus, you will never have one data vector with five significant digits and another with nine). This not only makes it a standard format for all components to read, it also makes it much easier for the user to read, as all data lines up cleanly into even rows and columns.

Although there is no specific format required for raw data before it is loaded into the spreadsheet, the user will need to transform the data into the following format:

ns, nd				
1	$d_{1,1}$	$d_{1,2}$	...	$d_{1,nd}$
2	$d_{2,1}$	$d_{2,2}$	...	$d_{2,nd}$
		.		
		.		
		.		
3	$d_{ns,1}$	$d_{ns,2}$	...	$d_{ns,nd}$

where *ns* is the number of data samples in the file, *nd* is the number of descriptors per data sample, and $d_{n,m}$ is descriptor number *m* for data sample *n*. It is not expected that this data format will change in future releases of XFCV. XFCV will add the sample # and count of row/columns to a raw data when using the spreadsheet function. These fields are required by XFCV for proper operation.

1.4 Preparing to use XFCV

After XFCV has been installed as per instructions in *XFCV Installing Guide V2.1* for the platform of interest, follow these steps to prepare to use XFCV:

For VAX/VMS users:

Add the following line to the file LOGIN.COM for each user running XFCV:

$ @*XFCVHOME*:XFCV_SETUP

where *XFCVHOME* is the location where the XFCV package has been installed (usually something like DKA300:[XFCV]).

For Unix (Ultrix, SunOS) users:

Add the following line to the file .cshrc for each user running XFCV:

***XFCVHOME*/bin/xfcv_setup**

where *XFCVHOME* is the location where the XFCV package has been installed (usually something like /usr/local/xfcv or /home/local/xfcv).

The setup procedure defines the locations for various XFCV components as well as user interface options. All users should read *XFCV V2.1 Release Notes* for specific notes regarding XFCV on the platform they are using prior to using XFCV for the first time.

Invoke XFCV on all platforms by entering the following command at the system prompt:

xfcv

You should see two windows appear, similar to those shown in Figure 1-1.

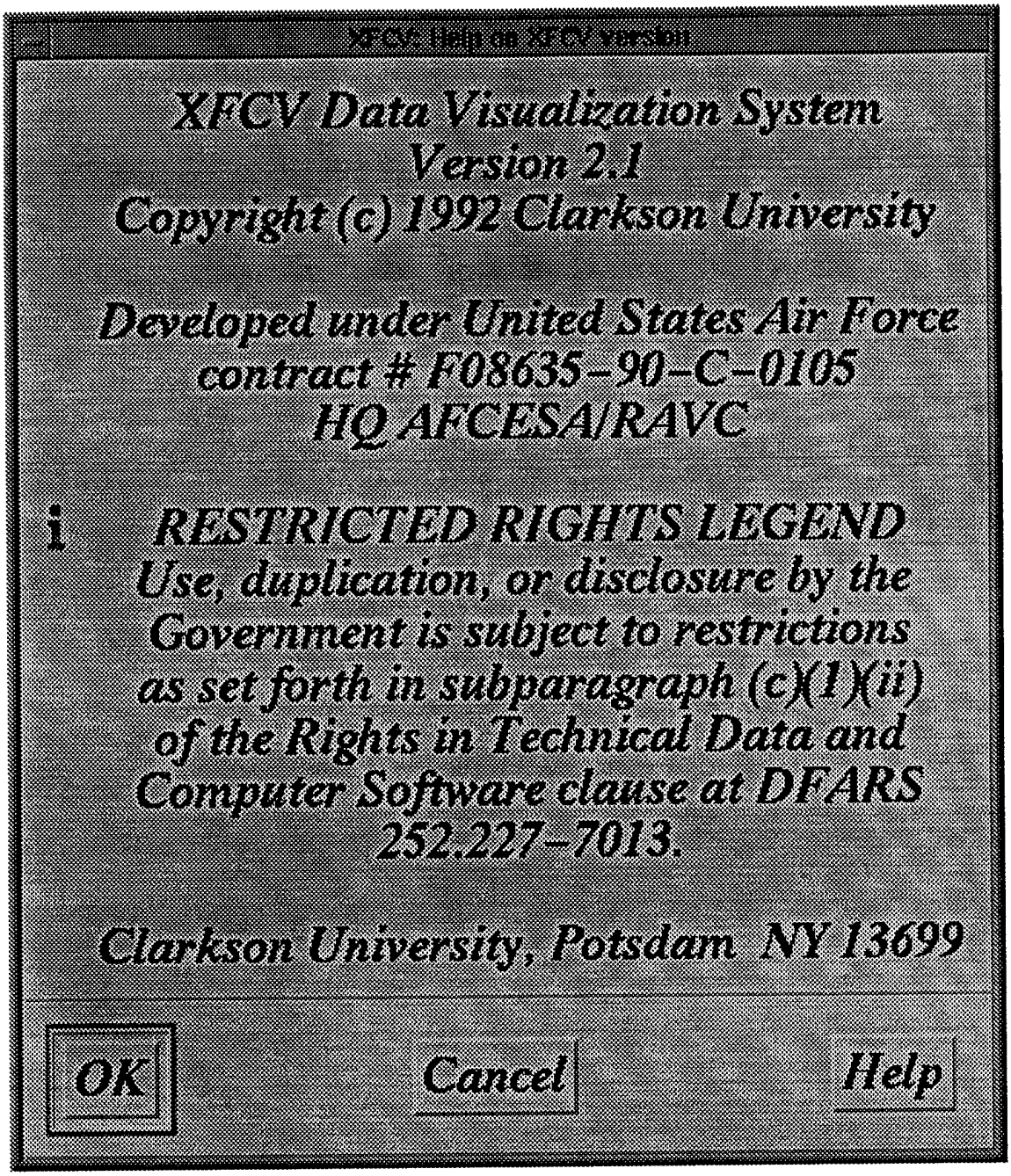


Figure 1-1 Startup Screen

Chapter 2

Overview of XFCV functions

This chapter details the functions of XFCV. Each pull-down menu in the system is shown and its functions discussed in detail.

2.1 General Notes on the XFCV User Interface

There are some general notes of interest for users who may not be familiar with OSF/Motif or MS-Windows user interfaces used by XFCV. If you have not used the windowing system before, you are advised to refer to the *OSF/Motif Style Guide* and/or the *OSF/Motif User's Guide* for Motif. These manuals describe the general appearance and behaviour (also known as *look and feel*) of the graphical user interface (*GUI*).

You will note that when you first start the program, you will see some items in the various menus are a lighter color than others. You will also find that these items are not selectable if you try to click on one. These items *greyed-out* because they have no meaning at this point in the system. For example, in the **File** menu, until you have loaded a PC coordinate plot file, there is no meaning to loading a class membership data file. Until the PC coordinates have been loaded, the class membership data has no data sample to be associated with. You will notice this greyed-out appearance elsewhere in the system when some item it not selectable at that point in time.

In addition to items which are greyed-out because of they have no meaning at a point in time, there are some functions greyed-out because they are not yet implemented, but have been put in the menus because the functionality is expected to be implemented in the V2.x timeframe.

2.2 File Menu

The file menu, as shown in Figure 2-1, contains selections to perform functions such as loading/saving different data files which comprise a data set, print the display or portion of it, and exit the system.

All load/save selections will popup a similar looking window. Take care to note the filename extension shown (also known as the *filter* in the file selector window) and the title on the window. You can readily determine the type of data file you are expected to select based on this information. It is possible to change the filter to select other files, but note that if the system is expecting

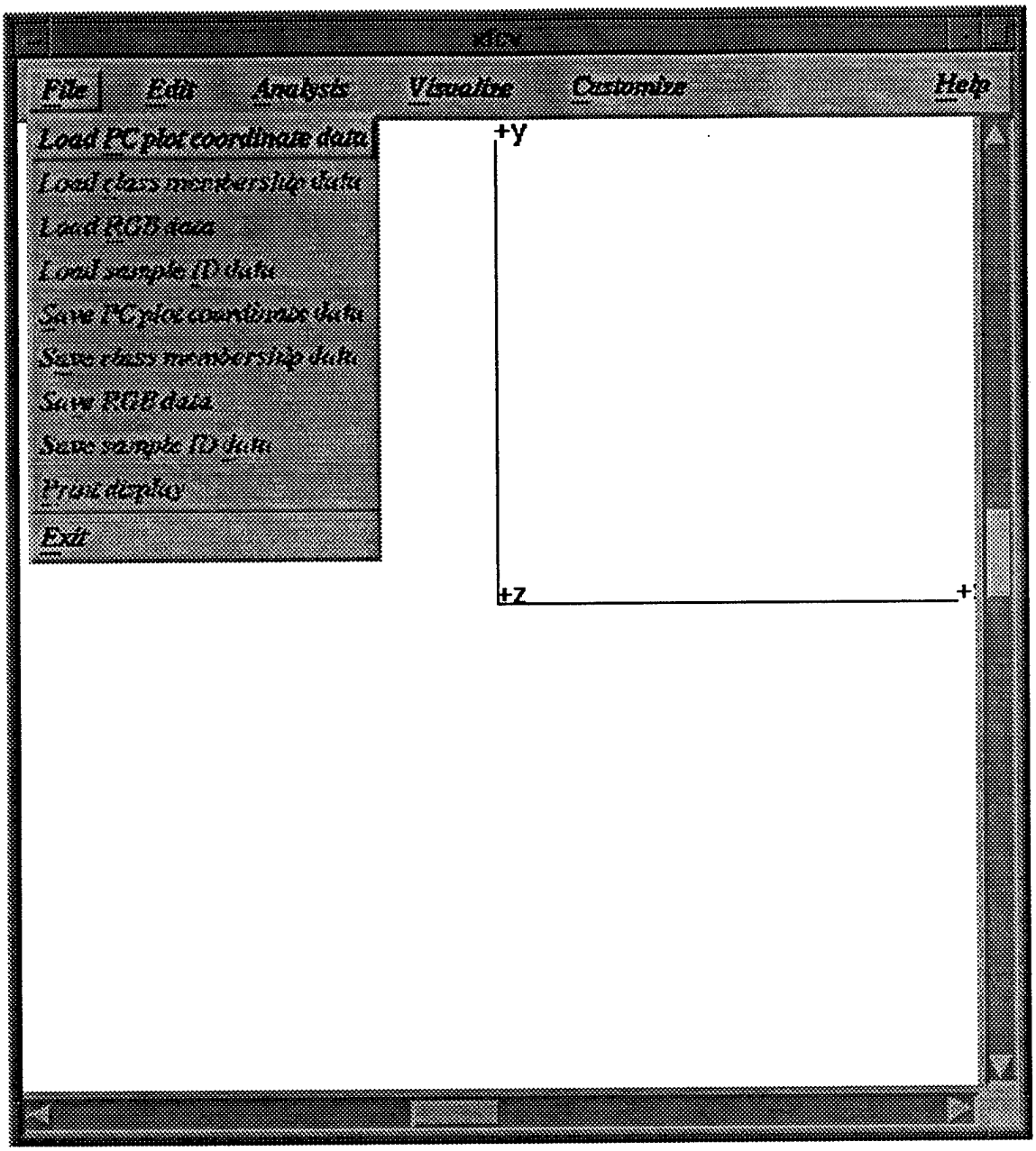


Figure 2-1 File Menu

you to load a PC coordinate data file and you load something else, unexpected results may occur.

Each file selection window is divided into five regions. The **Filter** string specifies some combination of a wildcard character along with the directory and file extension. The filter is a limiting function, which reduces the number of files which appear in the **Files** list. For example, for the PC coordinate data files, the end of the filter will look like **.pcdata*. This filter will enable you to view only the files in the directory that you are interested in, rather than all of them. To select a file, move the mouse pointer (or cursor) to the desired file in the File region and either click the MB1 (mouse button 1) once to highlight it. To actually select a file to be loaded, you can either point at the name again and this time double-click on it or with the filename hightlight, select the **OK** buttom. If you do not wish to load any file, you can dismiss the window by clicking once on the **Cancel** button. Currently, the **Help** button in all file selectors does nothing. In a future release of XFCV, it will either be greyed out (become non-selectable) or will pop-up help relevant to the file selector in question.

2.2.1 Load PC plot coordinate data

This selection will popup a file selection window as in Figure 2-2. The window will only display PC coordinate data files (usually designated with the filename extension *.pcdata*) that exist in the directory. Once the set is loaded, all data currently displayed is replaced with the new data set. Any other data sets previously loaded (class membership, RGB data, data set description, etc) are also initialized.

If any of the previously loaded data has been modified since it was first loaded, you should save any changes you wish to retain or they will be lost once the new data set is loaded.

2.2.2 Load class membership data

This selection will display a file selector window allowing you to load a class membership data set corresponding to the PC coordinate data set previously loaded. Any previously loaded class membership data will be replaced with the newly loaded data.

If any of the previously loaded class membership data has been modified since it was initially loaded, you will be queried to save thechanges or allow the changes to be discarded.

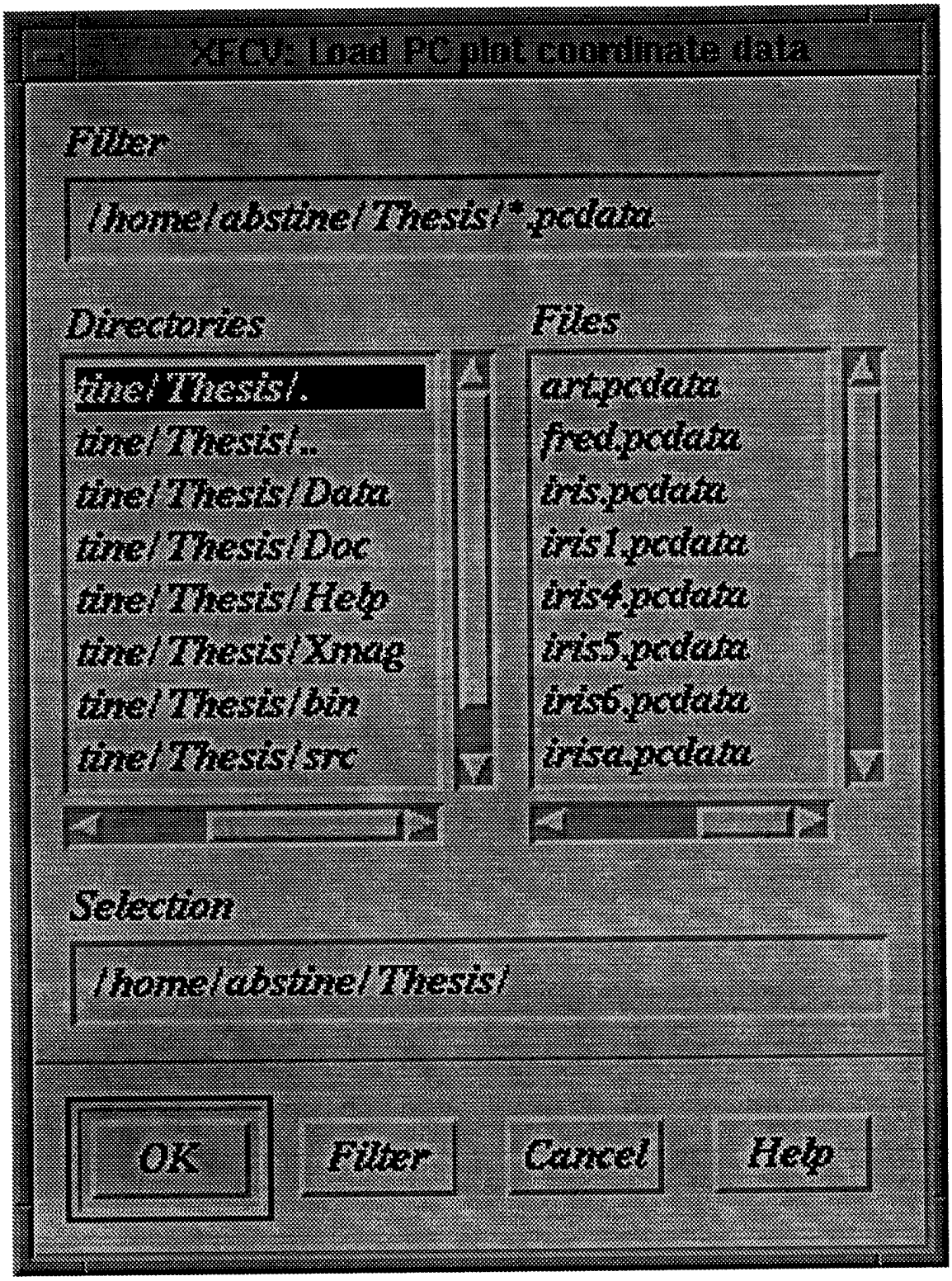


Figure 2-2 File Selector Pop-up

2.2.3 Load RGB data

This selection will display a file selector window allowing you to load a RGB (color) data set corresponding to the currently displayed PC coordinate data set. Any previously loaded RGB data set will be replaced with the newly loaded data.

If any previously loaded RGB data has been modified since it was initially loaded, you will be queried to save the changes or allow the changes to be discarded.

2.2.4 Load sample ID data

This selection will display a file selector window allowing you to load sample ID's corresponding to the currently displayed PC coordinate data set. Any previously load ID informationation will be replaced with the newly loaded data.

2.2.5 Save PC coordinate data

This selection will display a file selector window allowing you to save the current PC coordinate data set. No changes are made to the displayed data sets upon saving data.

2.2.6 Save class membership data

This selection will display a file selector allowing you to save the current class membership data set. No changes are made to the displayed data set upon saving data.

2.2.7 Save RGB data

This selection will display a file selector allowing you to save the current RGB data set. No changes are made to the displayed data sets upon saving data.

2.2.9 Save sample ID data

This selection will display a file selector allowing you to save the current sample ID's and assigned cluster information.

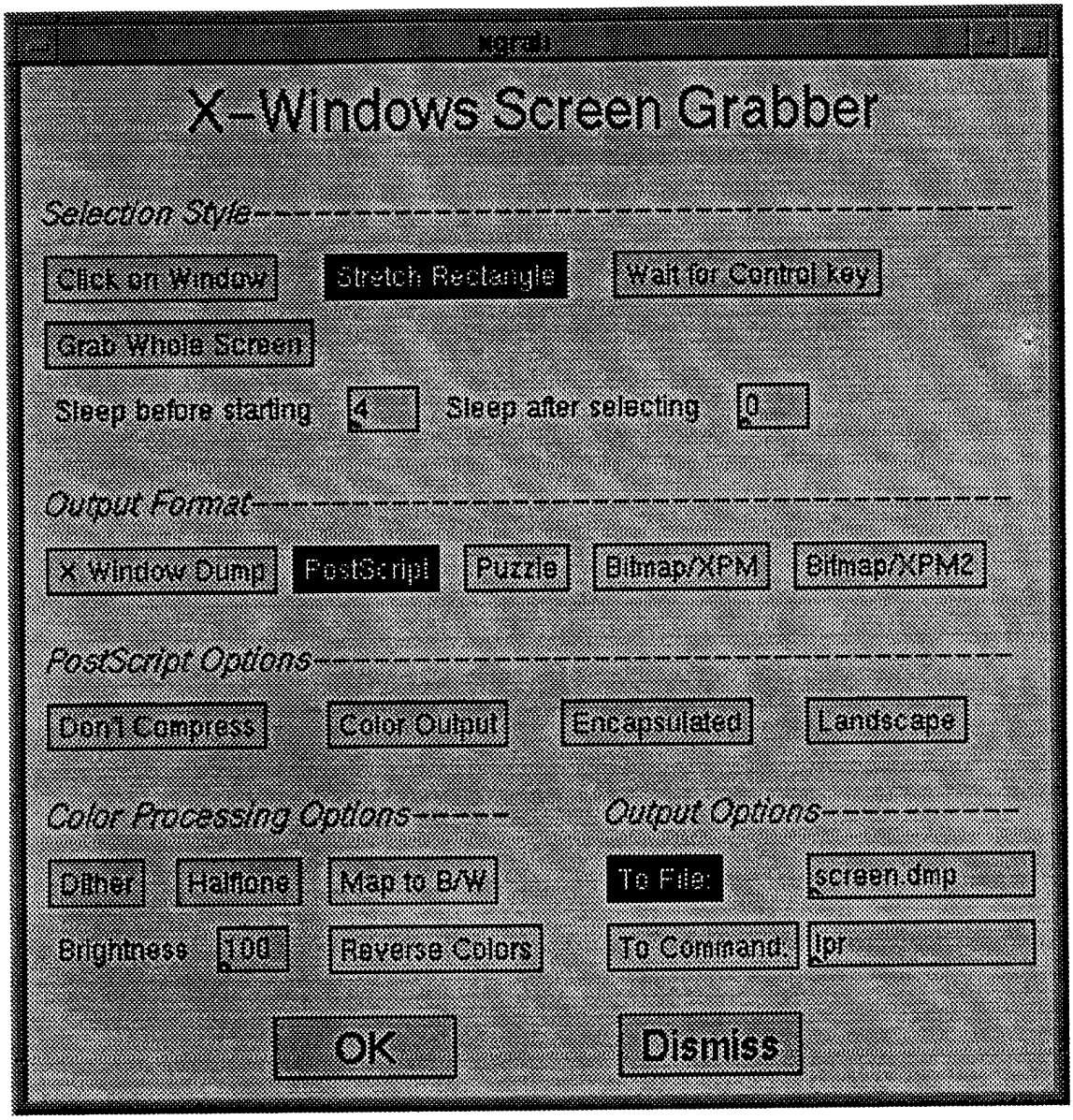


Figure 2-3 Print Display Pop-up

2.2.10 Print display

This selection will display a window, as shown in Figure 2-3, with various print options:

- Print entire display
- Print selected portion of display
- Save entire display to specified file
- Save selected portion of display to specified file
- Specify print format (PostScript, Color PostScript, X Window Dump (XWD), etc)
- Specify various image transformation (dithering, halftoning, gamma factor, etc)

XFCV Version 2.1 utilizes the public-domain programs xgrab and xgrabsc to perform its printing functions. You can refer to the Appendices for additional documentation on those components.

2.1.9 Exit

This selection will exit the XFCV system. You will be queried to confirm exiting the system via a pop-up. By selecting **OK**, you confirm exiting XFCV. You should verify that you have saved any changes to your data files before exiting. If you forgot to save something or inadvertently selected Exit, choose **Cancel**, and XFCV will not exit.

2.3 Edit Menu

The edit menu, as shown in Figure 2-4, has several options for editing and transforming data sets, including editing currently displayed data points, invoking the spreadsheet, and invoking the data scaling component. For V2.x, the data scaling and spreadsheet components are actually external program that are called up into a window via the menu. Depending on the platform, you may notice the main screen 'freeze' when one of these functions is active. Do not be alarmed - when you exit that component, the screen with come back to life.

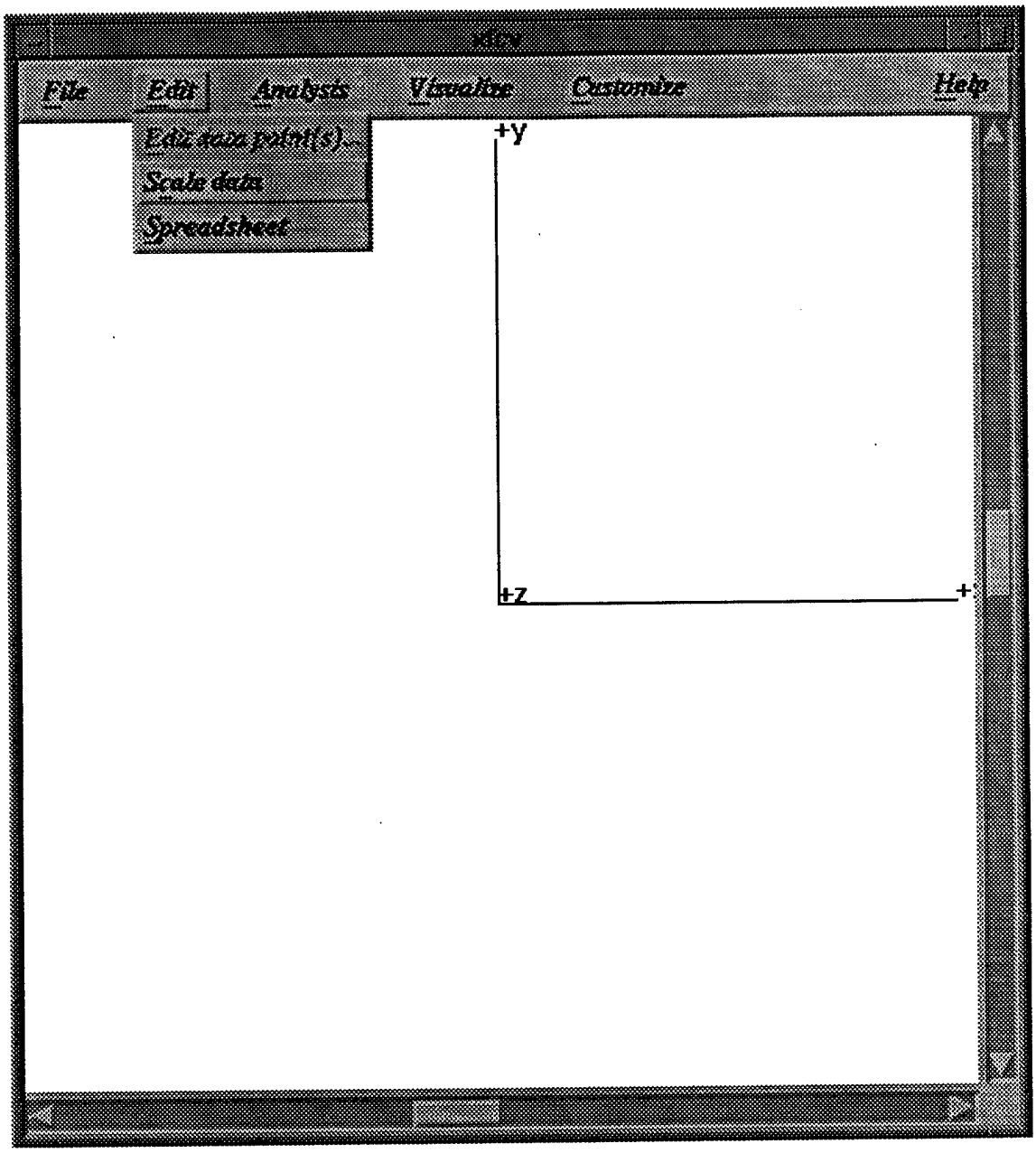


Figure 2-4 Edit Menu

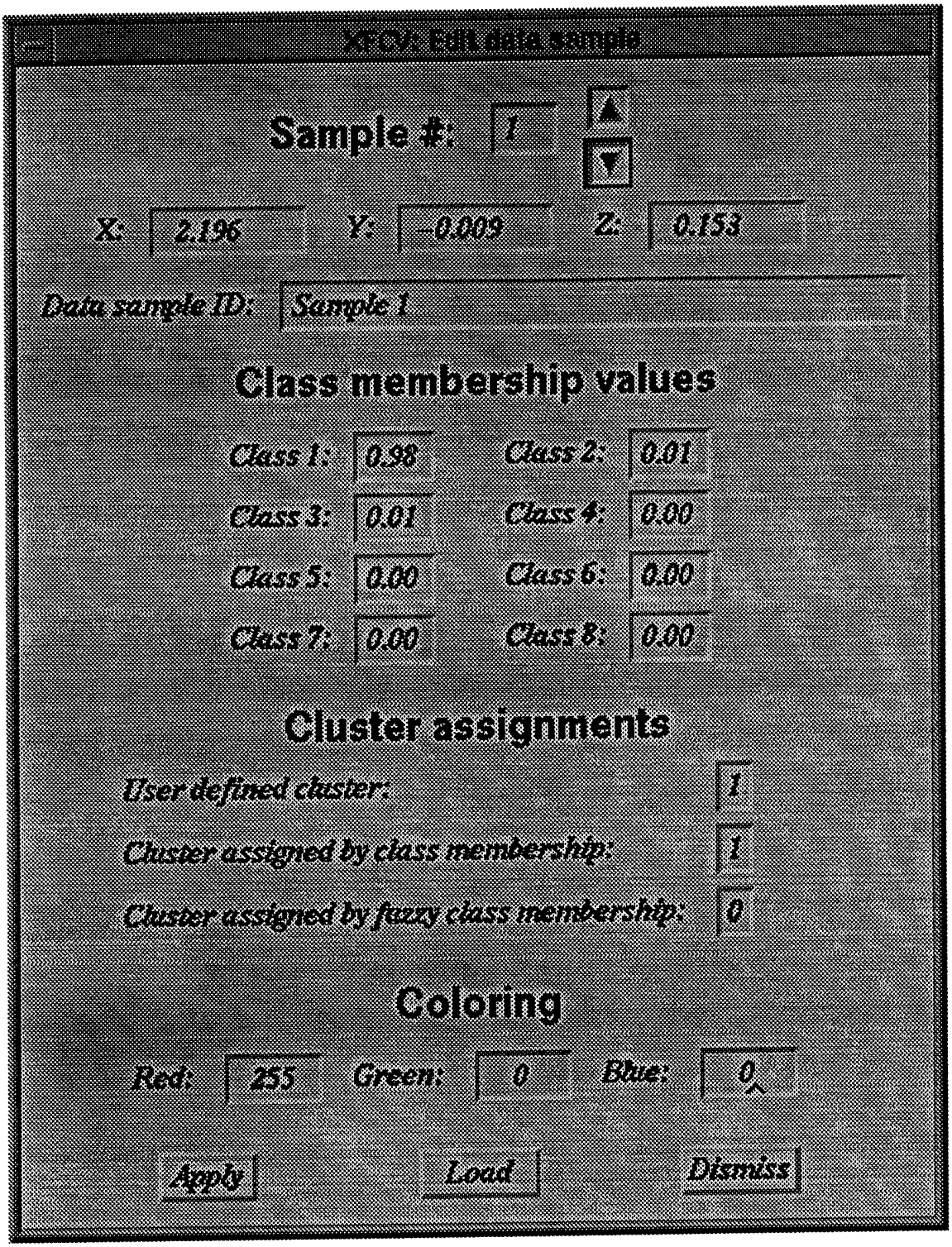


Figure 2-5 Edit Data Point(s) Pop-up

2.3.1 Edit Data Point(s)...

This selection will display a dialog window, as shown in Figure 2-5 which allows you to modify specific parts of a data sample, including its *x,y,z* coordinates, class membership data, coloring information, cluster information, etc. You will notice that the dialog window has most of its fields greyed-out. To select a data sample, enter its integer ID in the text field next to the **Sample #** prompt. Then, using the mouse point, click once on the load sample button. This will load the data sample's various components into the dialog windows fields and the fields will change from being greyed-out to the normal text color. Now you can modify the values of the parameters you wish. To make these values take effect, click once on the **Apply** button and the selected data sample will be updated with the new values. If you wish to discard any changes, click once on the **Dismiss** button and the dialog window will go away without changing any parameters.

2.3.2 Scale Data

This selection invokes the data scaling component of XFCV, *XFCV_Scaling*. This program allows you to perform various scaling functions on a raw FCV data set, prior to analysis via *XFCV_Clustering* or *XFCV_PCPlot*. You can perform various scaling on the data including: autoscaling, normalization, log-scaling, etc. as shown in Figure 2-6.

2.3.3 Spreadsheet

This selection invokes the XFCV spreadsheet. A file selector will be displayed, allowing you to specify the file to be edited with the spreadsheet. The default file type for editing is *.dat*, but you can change this by editing the *Filter* specification in the file selector. Once you have choosen a file, you will be presented with a dialog box, which will you can use to select the type and format of the file, as shown in Figure 2-7. Currently, you can choose to edit FCV data files and sample ID files. Note that sample ID files will use a standard text editor for editing the file. For FCV data files, you can select the format of the data file, if you know. The choices are:

- Raw - the file has no row labels, no row/column counts at the top
- Row labels only - the input file only has row labels and still needs row/column counts added.
- Row/Column count only - the input file only has row/column counts and still needs row labels added.

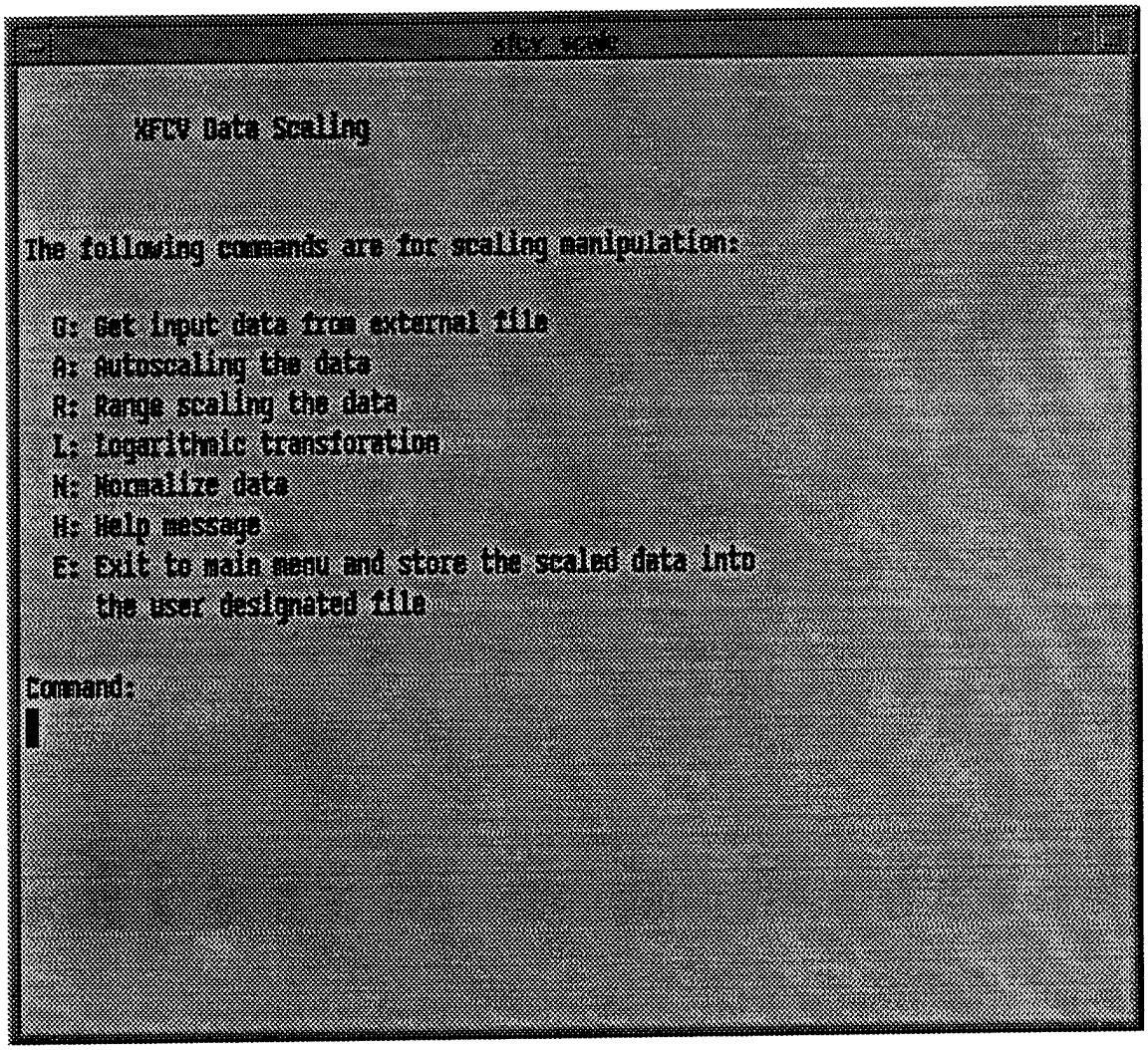


Figure 2-6 Scale Data Pop-up

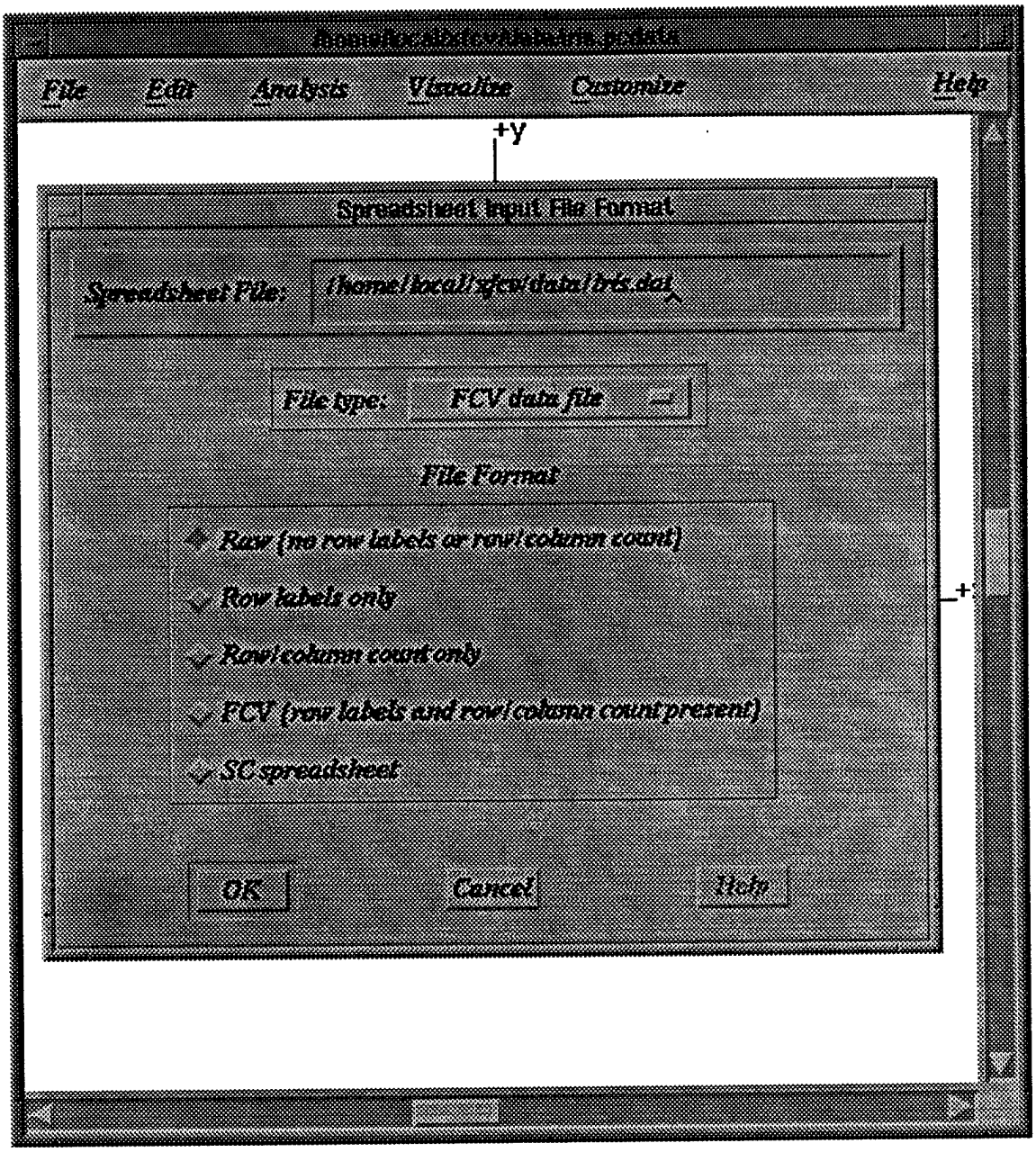


Figure 2-7 Spreadsheet Format Dialog

- FCV - input file has all necessary formatting.

- SC - the input file is an internal sc spreadsheet file and needs no additional formatting.

If in doubt, elect to use *Raw*, and once you are in the spreadsheet, you can remove any redundant fields (ie, its easier to remove the fields than it is to add them by hand). In V2.x, the public domain spreadsheet *sc*, as shown in Figure 2-8, is used on the various workstation platforms (VAX/VMS & Unix). For documentation on using the sc spreadsheet, refer to Appendix A.

2.4 Analysis Menu

The Analysis menu, as shown in Figure 2-9, has allows you to select among the analysis components of the system including: Principal Component Projection (PCPLOT), FCV cluster analysis, and others. This menu may have other components in it on different platforms, as some additional analysis components may have been added.

2.4.1 Perform Principal Components Projection

This selection invokes the XFCV component program *XFCV_PCPlot*, to perform principal components projection of the raw FCV data set specified. When invoked, this component pops up in its own window. For V2.x, this component is not a GUI based module, but rather uses a simple text based interface (using the supplied xterm). The PC plotting component is shown in Figure 2-10.

2.4.2 Invoke FCV analysis

This selection invokes the XFCV clustering analysis component, *XFCV_Clustering*, as shown in Figure 2-11. As with the PC projection module, the FCV clustering component also has a text based interface.

2.4.3 Invoke Projection Pursuit analysis

This selection invokes the XFCV component program *XFCV_ProjectionPursuit* to perform Projection Pursuit method projection of the specified raw FCV data set.

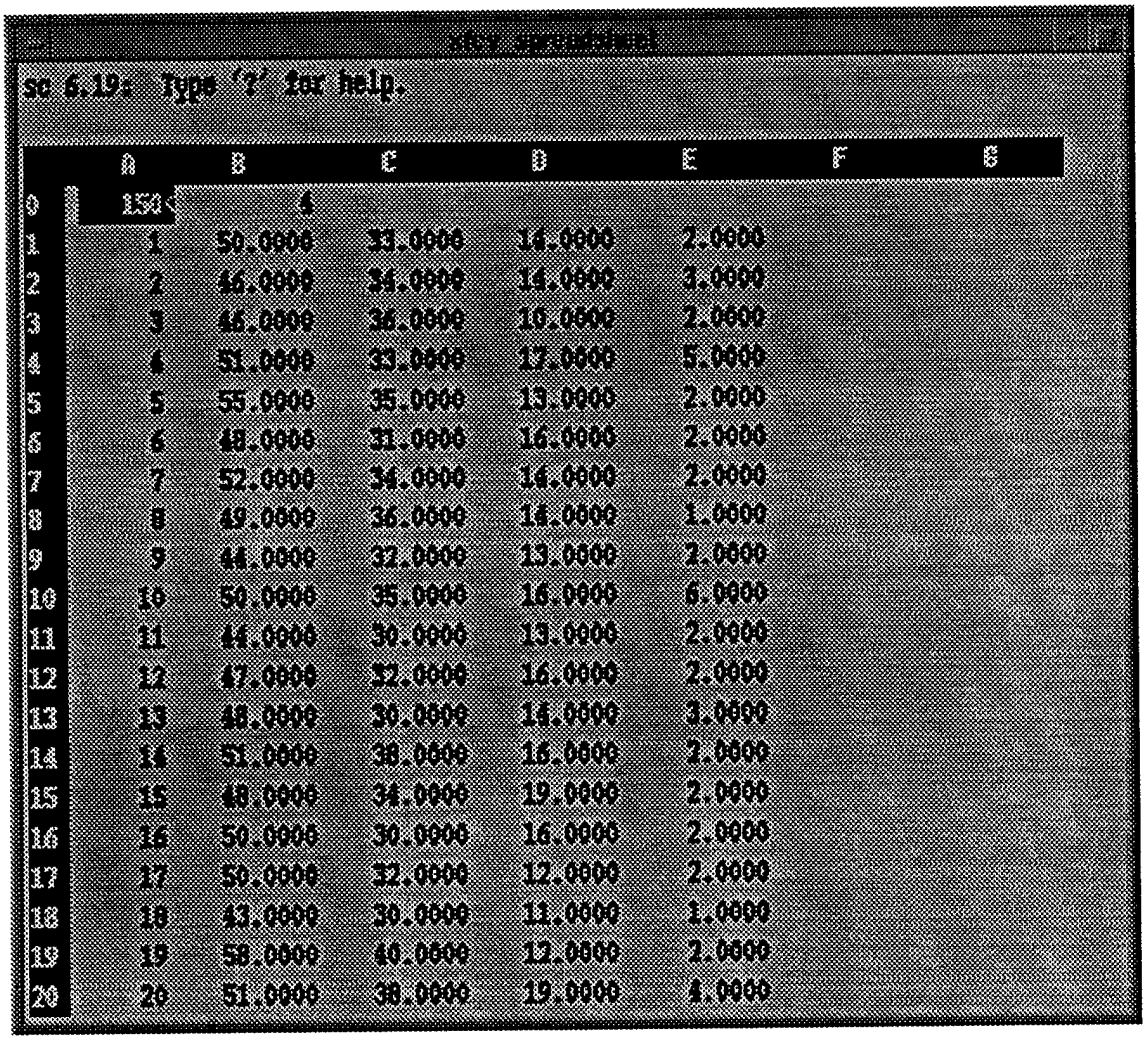


Figure 2-8 Spreadsheet Pop-up

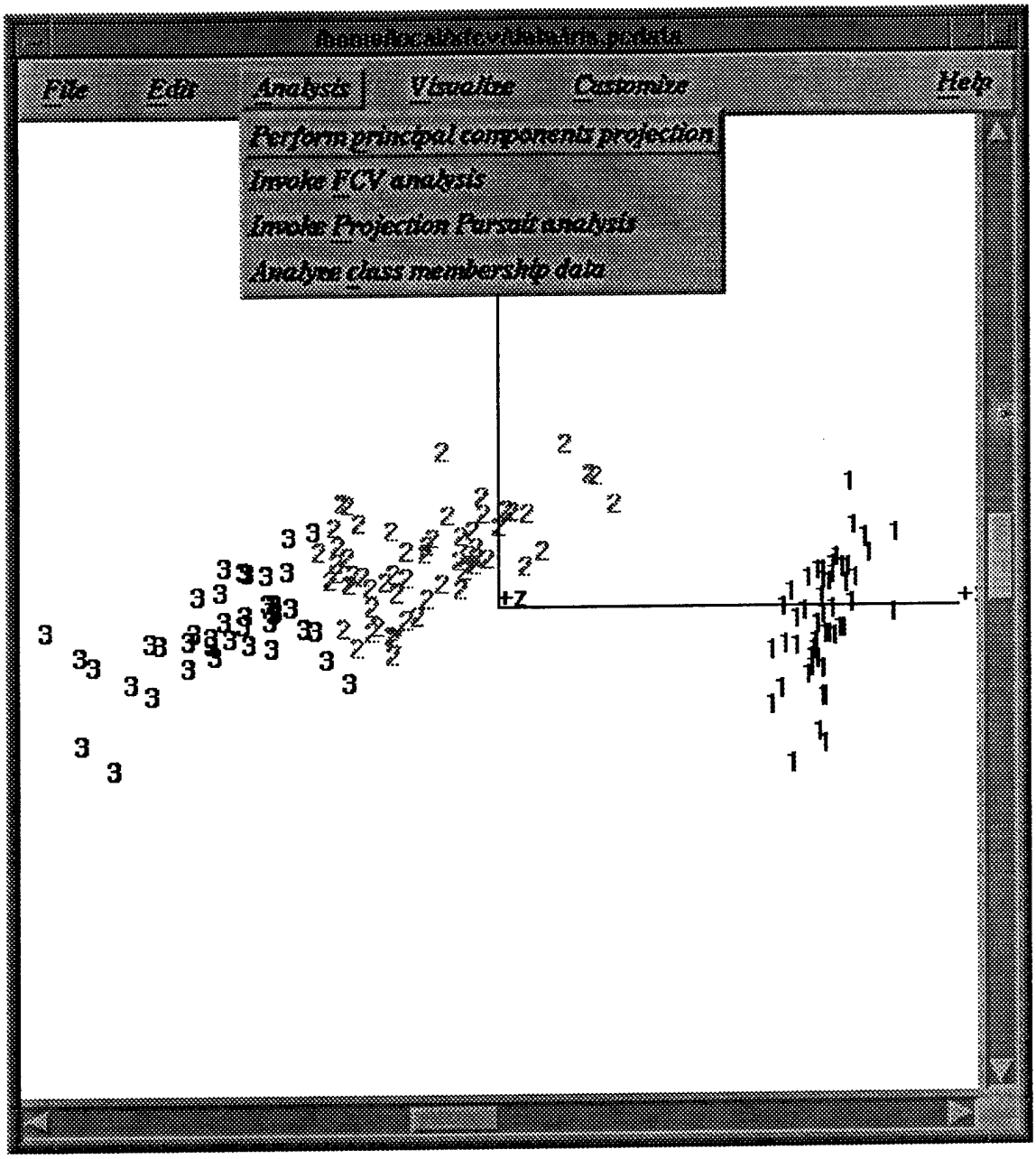


Figure 2-9 Analysis Menu

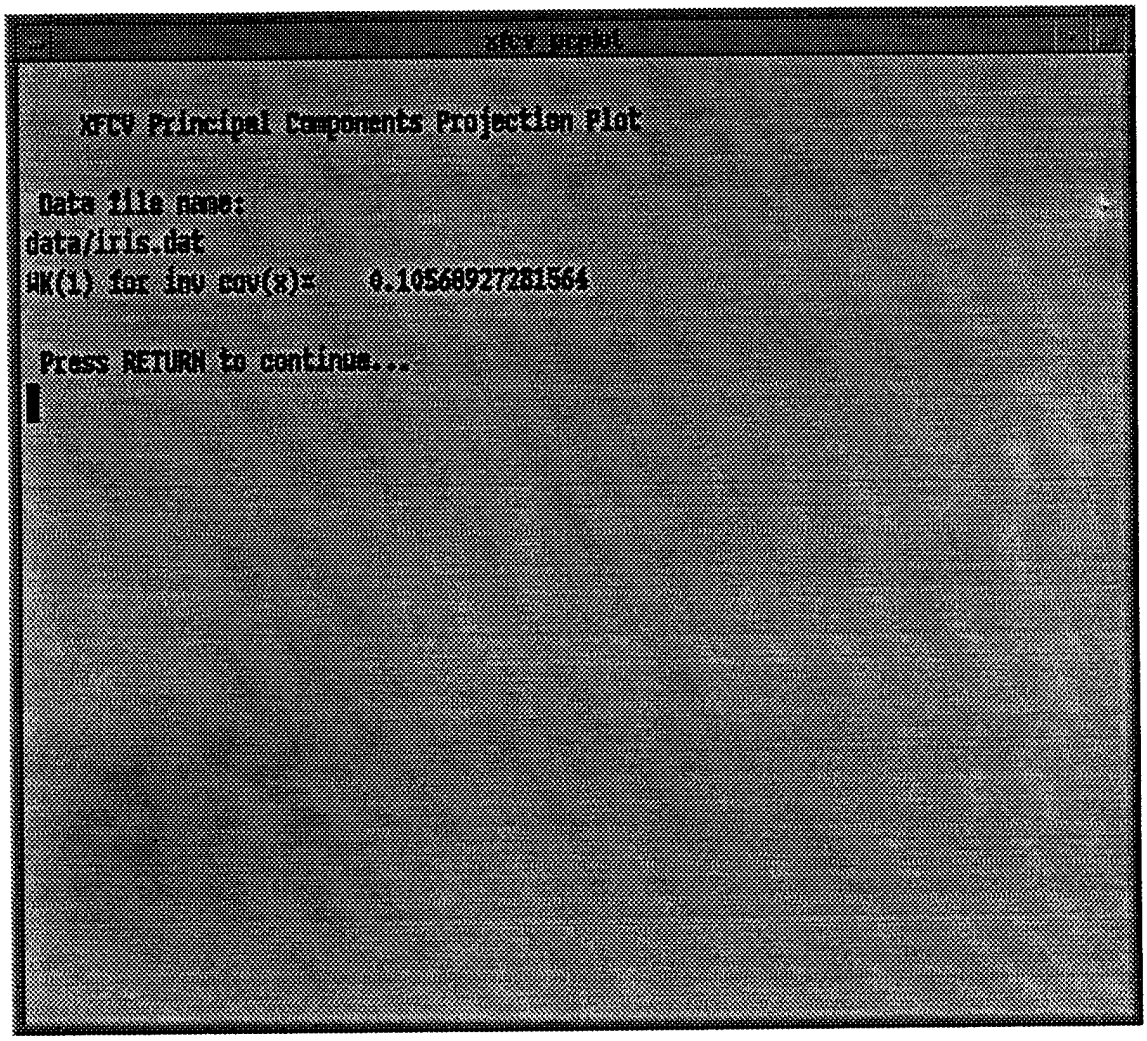


Figure 2-10 Principle Components Plot Pop-up

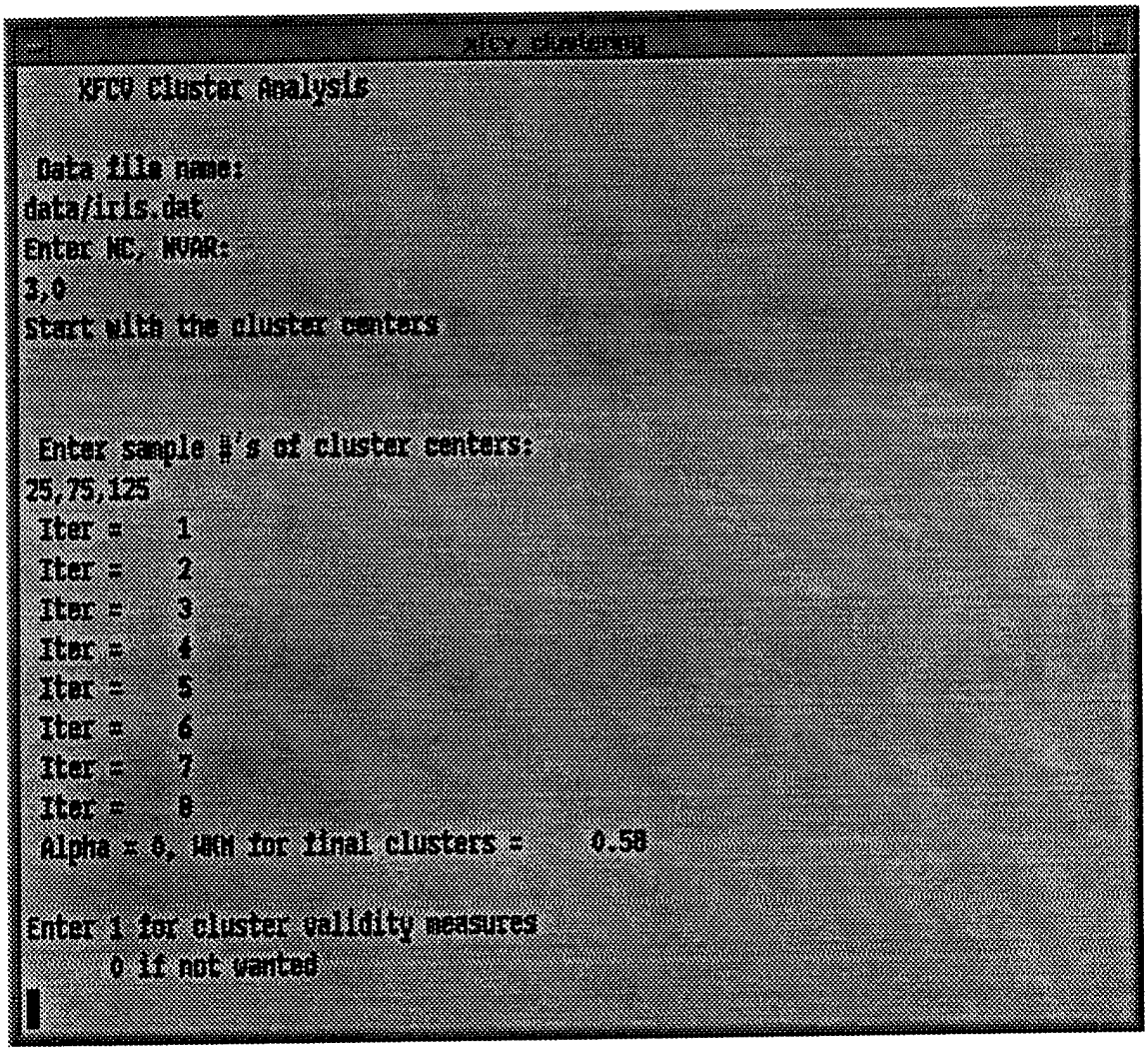


Figure 2-11 Invoke FCV Analysis Pop-up

2.4.4 Analyze class membership data

This selection invokes a user-defined class-membership/color analysis component to perform additional analysis of class membership data for purposes of specific color definition.

2.5 Visualize Menu

This menu, as shown in Figure 2-12, has several options allowing you to affect the visualization of the data samples, including coloring, axes selection, *x,y,z* rotations, zoom factor, etc.

2.5.1 Visualization options...

This selection invokes a dialog to allow you to select various visualization methods as shown in Figure 2-13:

- *Sample point markers*
  - Show cluster/class #
    This selection will display the data points according to their cluster. Further specification of the cluster # is available with addition options.
  - Show sample point #
    This selection will display the data points according to their ID as assigned in the PC plot data file.
  - User Defined Marker
    This selection will display the data points by a user-defined marker (refer to customization options for definition of the markers)
- *Sample cluster/class # assignment*
  - By largest member #
    This selection will display the data points according to their largest class membership value. The marker displayed will be the number class #.

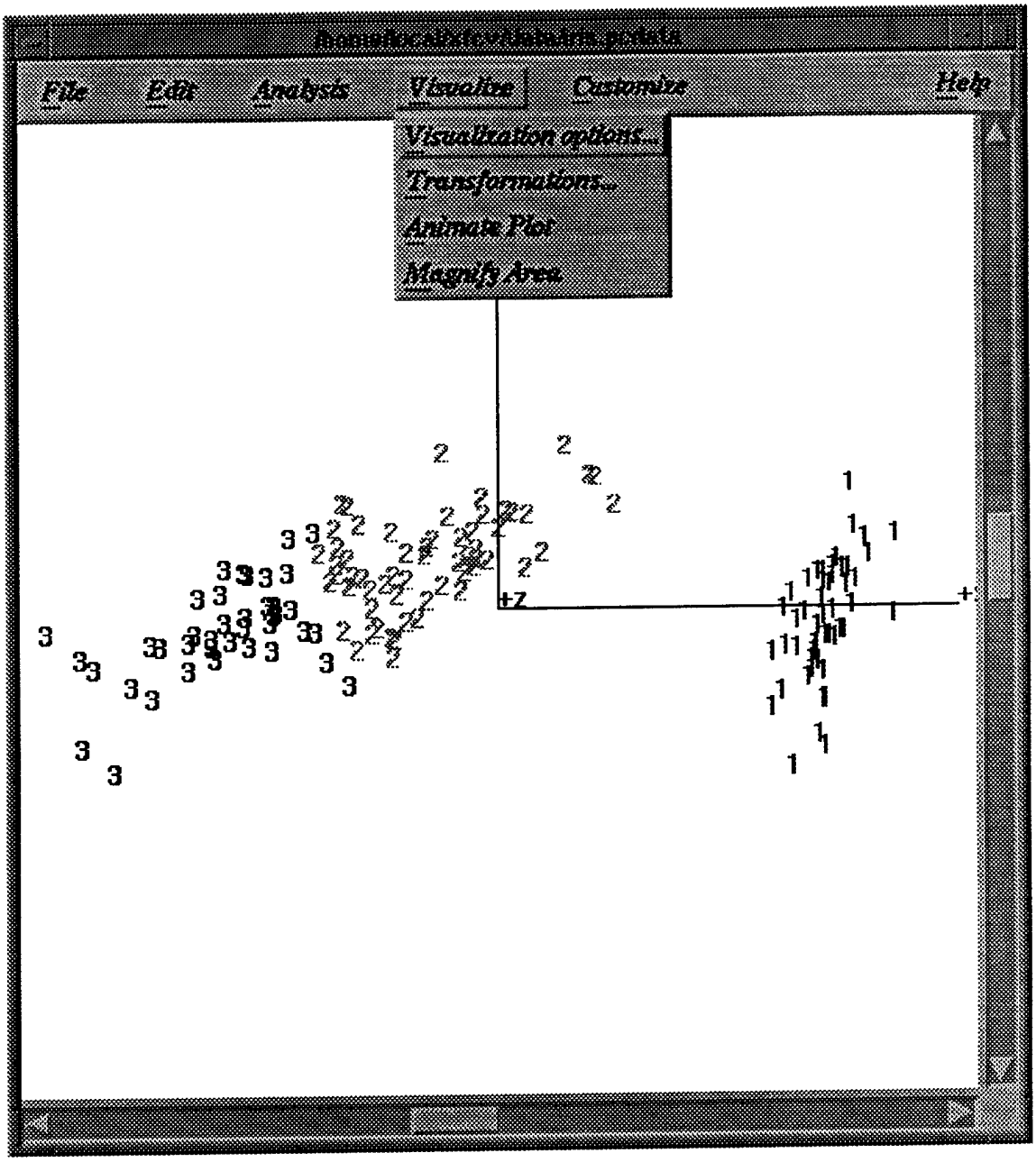


Figure 2-12 Visualize Menu

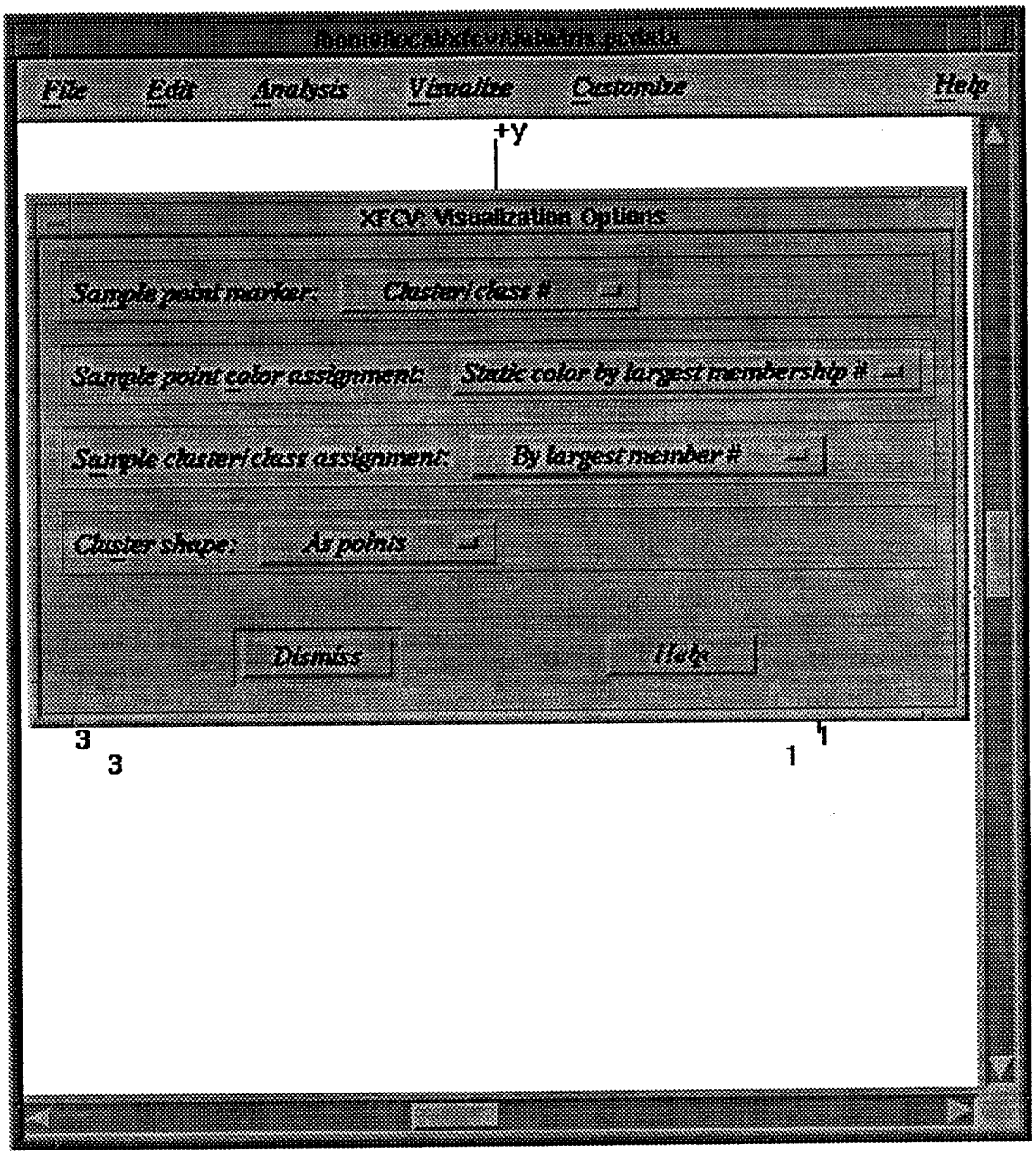


Figure 2-13 Visualization Options Pop-up

- As pre-defined in data set
  This selection will display the data points according to the cluster definitions in the data set.

- As combination (fuzzy)
  This selection will display the data points by their largest class membership and where a point has two membership value within the defined 'fuzz' factor, the point will be displayed as "n/m", where n is the largest member class and m is the second largest member class. This is useful to see points which have close membership values in two classes.

- *Sample point color assignment*

  - No Color (b&w)
    The data samples displayed will only be displayed in the foreground color (e.g., black. No color information from the class information or RGB database will be used.

  - Static Color by Assigned Cluster #
    The color of the data samples will be displayed according to the cluster/class to which they are assigned.

  - Static Color by Largest Membership #
    The color of the data samples will be displayed according to the cluster/class in which the sample has the highest membership.

  - Static Fuzzy Color
    The color of the data samples will be displayed according to the cluster/class in which the sample has the highest membership. Additionally, if a data sample has a high membership value in two classes (within the defined fuzzy color tolerance, default=0.05), then that data sample will be displayed with a color that is a straight linear mix of the colors of the those two classes.

  - Dynamic Color
    The color of the data samples will be a function of the class membership values. For example, if the point has a membership value of 60% in class 1, 20% in class 2, and 20% in class 3, then the sample will be colored 60% x (color of class 1) + 20% x (color of class2) + 20% x (color of class3).

This gives you the effect of a probability density function with regards to the class membership values and the sample's coloring. The visual effect should be that points that are further away from the cluster center will be less like the color of the center and more like either a neighboring cluster or just darker.

- Use Color as Defined in RGB File
  The color of the data samples will be display according the the RGB data that was loaded for this data set. If no RGB data has been loaded, the samples will be colored the foreground color (black).

- *Cluster Shape Visualization*

  - As Points
    The data samples will be displayed as points in space (with markers as previously specified.

  - As Starburst
    The data samples will be displayed at points in space (with markers as previously specified). In addition, each point will be connected with a ray from the cluster center. The effect of this will be that the shape of the cluster will be more apparent.

2.5.2 Transformations

This selection invokes a dialog allowing you to change the zoom factor, x,y,z rotations and axes to display as shown in Figure 2-14.

2.5.3 Animate Plot

This selection initiates an animation of the display, performing rotations and zoom factor as defined by the *Customize Animation Parameters* dialog.

2.5.4 Magnify Area

This selection will bring up a window which will allow you to select (grab) an area of the display and magnify it (on a pixel level). This is useful when trying to discern the subtlies of the coloring of various samples. For V2.x of XFCV, the public-domain xmag (Motif version) is used. To avoid naming conflict with the standard X magnifier, xmag, the XFCV magnifier has been renamed

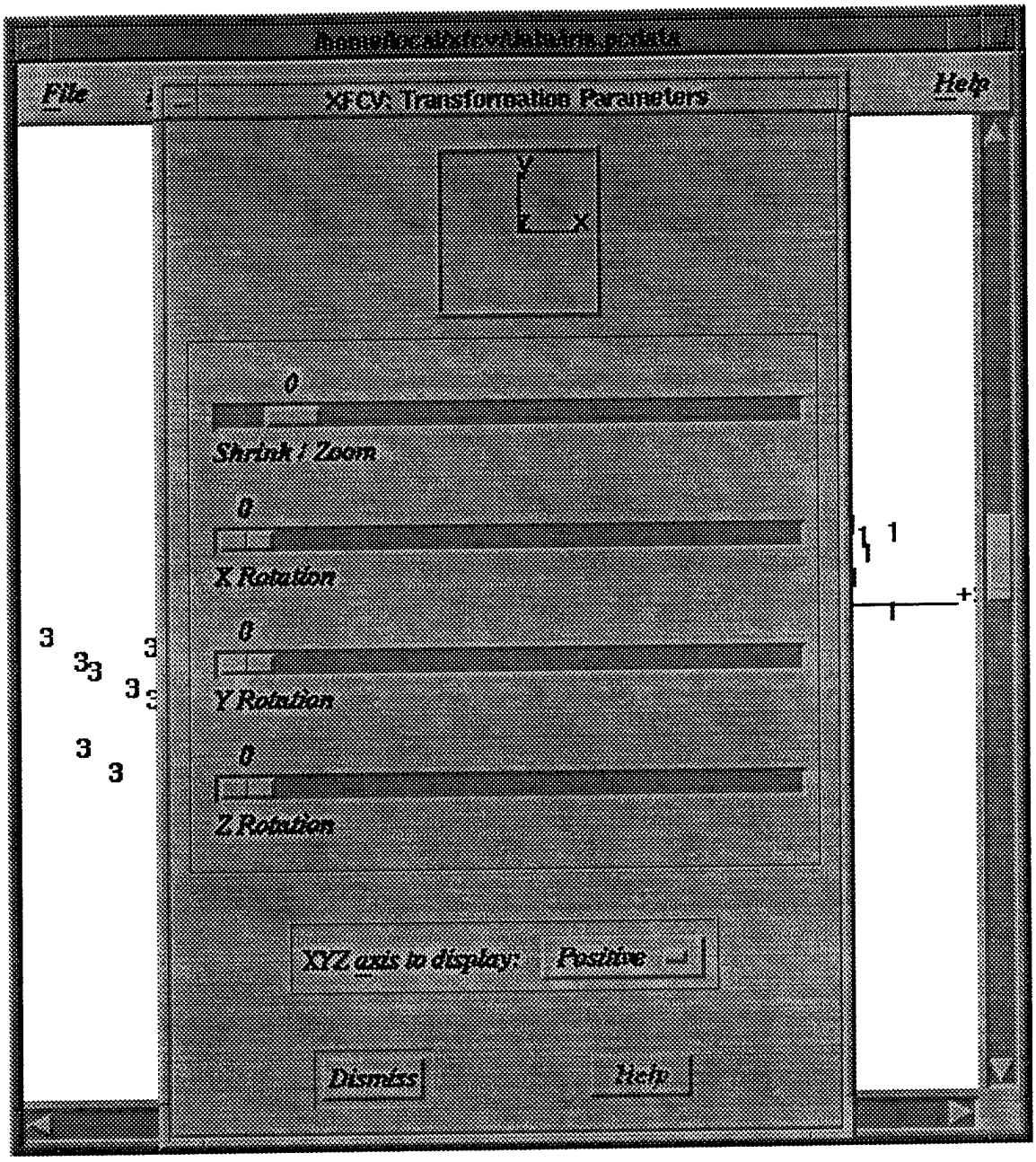


Figure 2-14 Transformations Pop-up

mmag. See Appendix E for documentation on using the magnifier. An example of the magnifier in use is shown in Figure 2-15.

2.6 Customize Menu

As shown in Figure 2-16, the selections in this menu allow you to change the behaviour or parameters of the system, including the animation parameters, fuzz factor, cluster coloring, etc.

2.6.1 Animation Parameters

This selection invokes a dialog window, as shown in Figure 2-17, with fields available to change the rotations, zoom factor, and number of frames to perform for the Animate Display function.

2.6.2 Visualization Parameters

This selection, as shown in Figure 2-18, invokes a dialog window which allows you to customize the settings of cluster coloring, fuzzy color tolerance, cluster markers, and dynamic coloring policy. Currently, the following fields are cluster specific and can be changed for each cluster/class:

- Base Cluster Color
- Cluster Marker

The other fields, *Fuzzy color tolerance* and *Dynamic coloring policy* are global system values and do not change for each cluster.

To use the dialog to customize parameters for a particular cluster/class, enter the cluster number in the *Cluster* # field (or use the up/down arrows to select the cluster number). To see the current parameters for that cluster, select the *Load* button. The color chip will show the color currently assigned to that cluster and the sliders will be set to the current RGB values for that color. Likewise, the other cluster specific fields will be displayed for the selected cluster. To modify the color of a cluster, move the RGB sliders to the desired values. You will note that when you modify the color of a cluster, its on-screen color will be updated automatically. This is done to allow better feel for the incremental modification of the coloring. To make a color change permanent, select the *Apply* button. This will update the cluster's parameters with all changes to cluster specific changes and global parameters you have modified. To cancel the changes, to can select to reload the previous values by selecting *Load* again or by exiting the dialog via the *Dismiss* button.

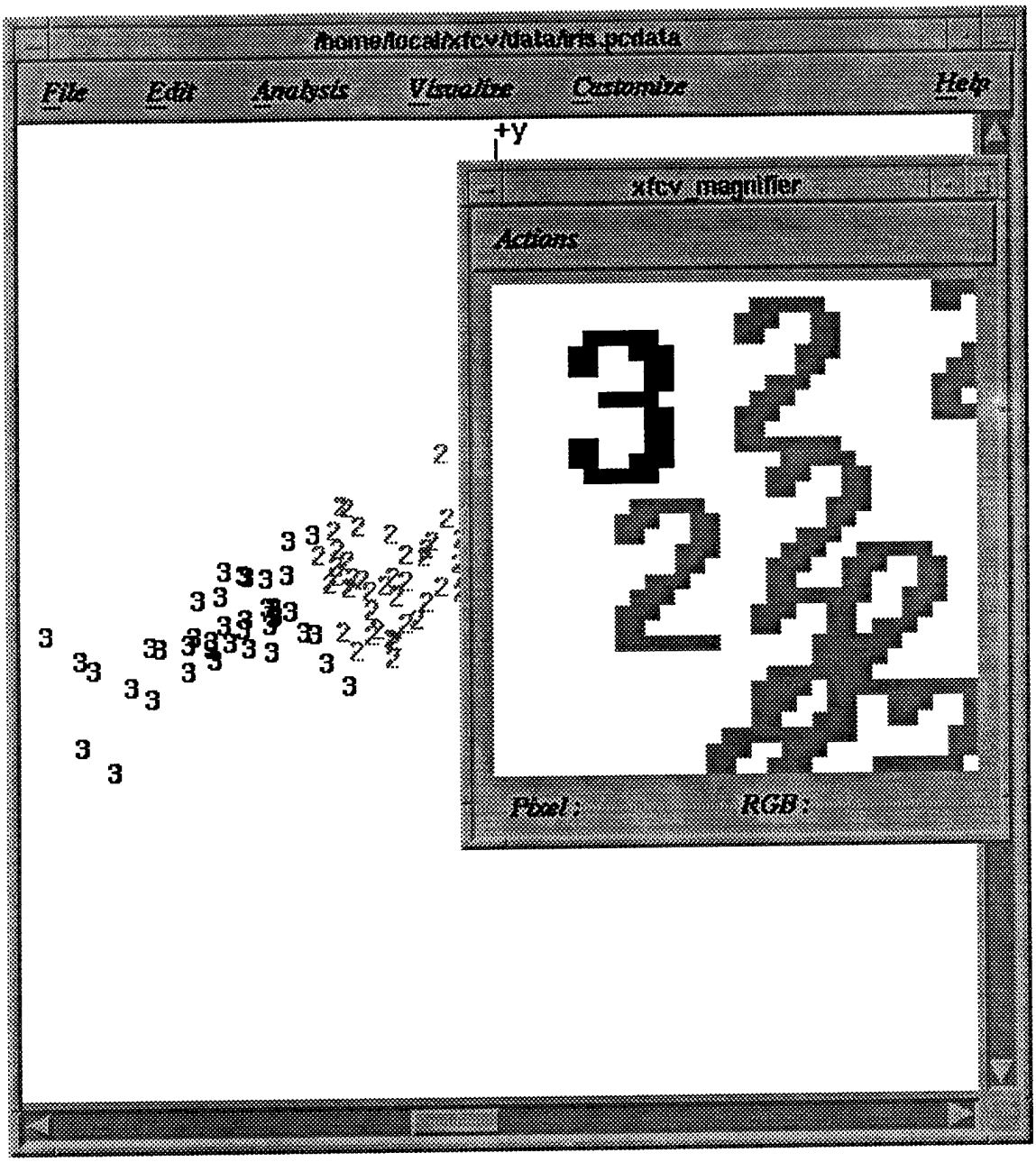


Figure 2-15 Magnifier Pop-up

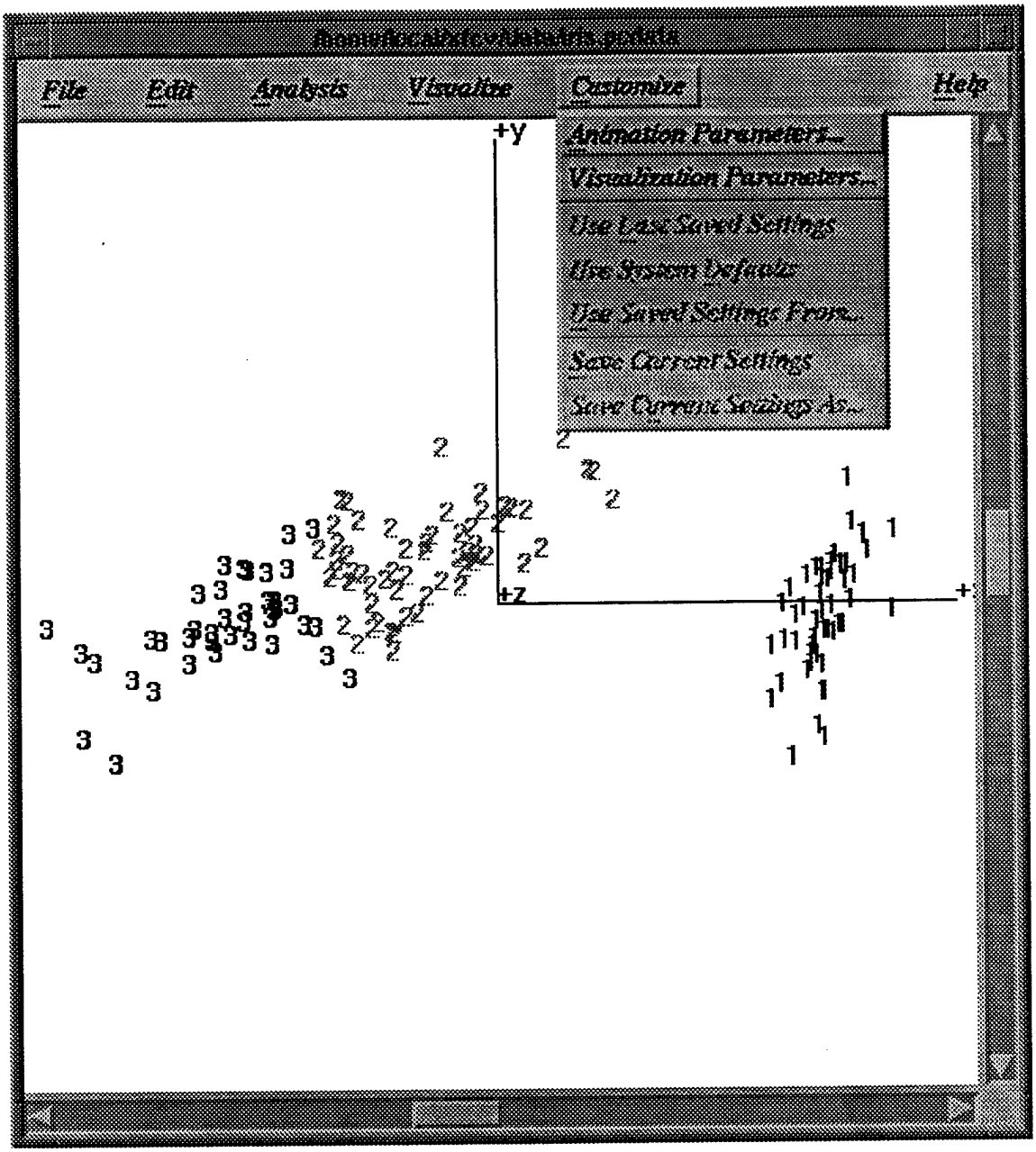


Figure 2-16 Customize Menu

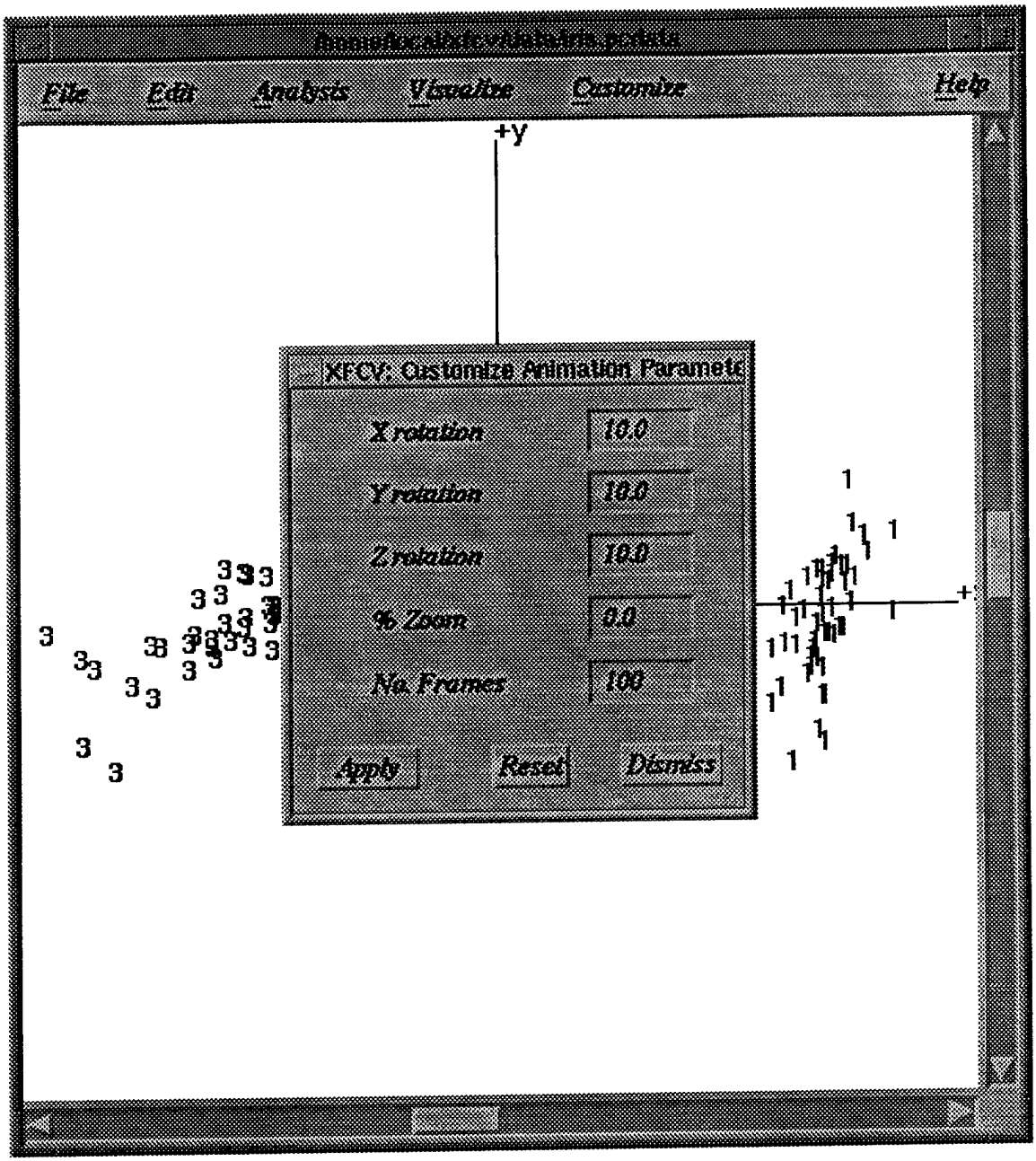


Figure 2-17 Customize Animation Dialog

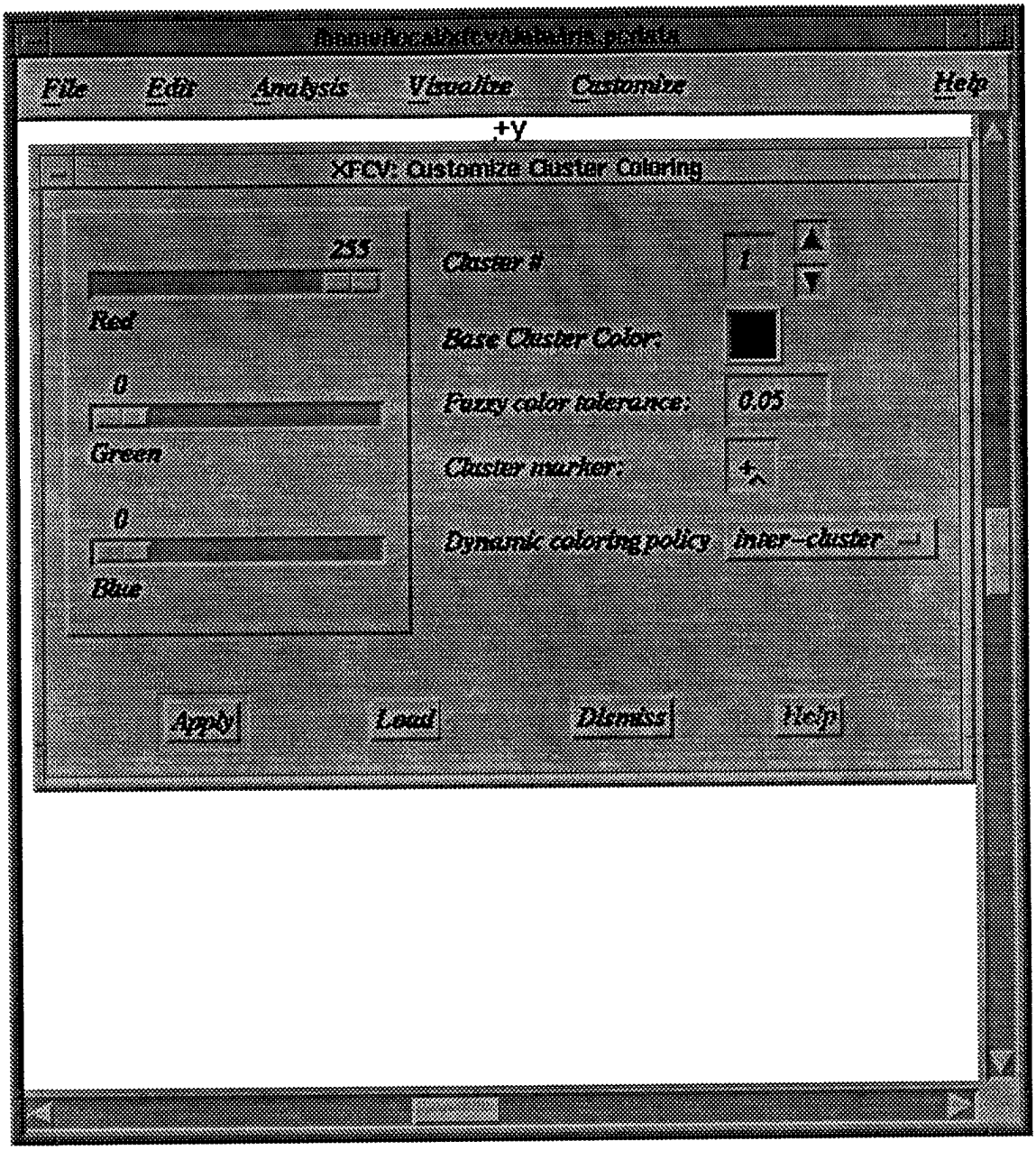


Figure 2-18 Customize Visualization Options Dialog

2.7 Help

This menu has a several selections for obtaining online help for XFCV, as shown in Figure 2-19.

2.7.1 On Help

This selection discusses the use of the online help system. It details the various different regions of the help windows, buttons, etc.

2.7.2 On Version

This selection pops up a single dialog box displaying the version, copyright, and licensing terms of the XFCV system.

2.7.3 On Tasks

This selection will pop-up a help window, as shown in Figure 2-20, allowing you to obtain help on each of the menu items in XFCV as well as performing various tasks.

2.7.4 On Release Notes

This selection will pop-up a help window allowing you to review the release notes for this version of XFCV. Included in the release notes are a brief overview of the new features in this release of XFCV as well as known problems and workarounds.

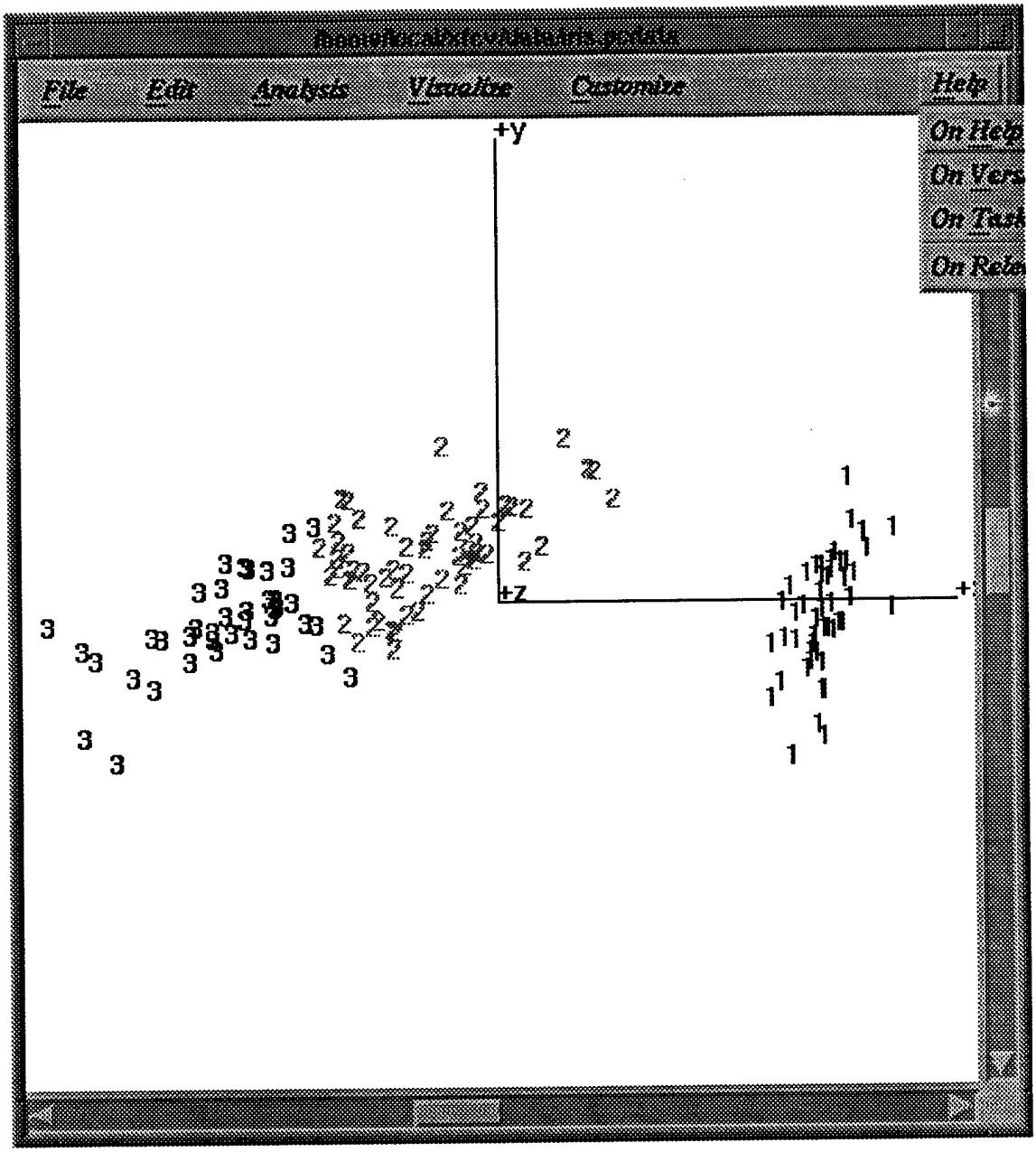


Figure 2-19 Help Menu

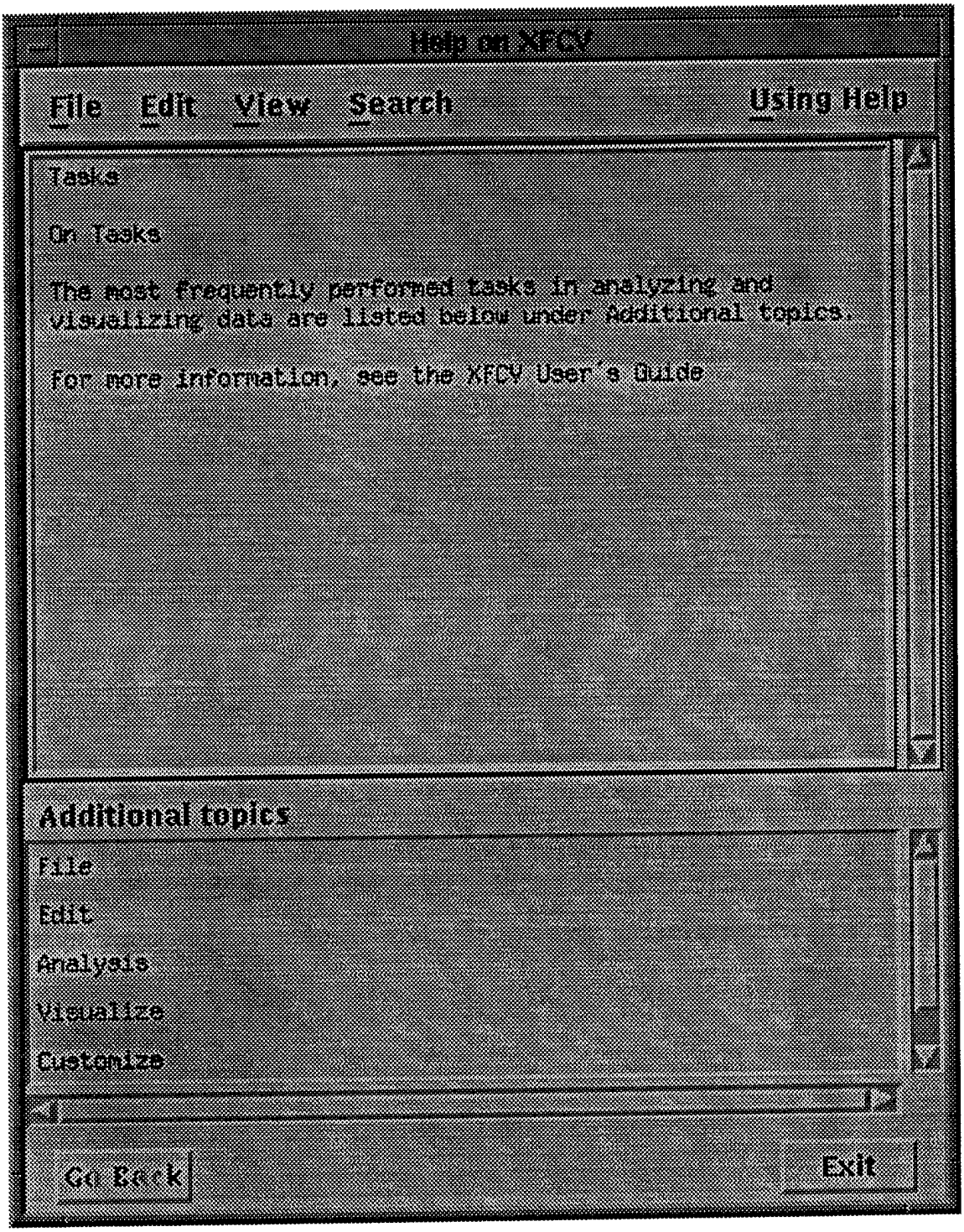


Figure 2-20 Help on Tasks Pop-up

Chapter 3
Performing Data Analysis and Visualization with XFCV

In this chapter we will perform the analysis and visualization of a sample data set. The data set we will use will be the IRIS data set. Each of the steps will be performed in the normal sequence you will use with other data sets. The IRIS data set is comprised of a set of iris measurements. It consists of four measurements on 150 flowers. The septal and petal lengths and widths were determined for 50 flowers of each of three varieties of iris. Thus our raw data set will be 150 samples of 4 descriptors per sample. In addition, there are three types of iris in the data set, which translates to three classes (clusters) we will be looking for in the analysis and display.

3.1 Data Entry

Data entry/editing for the XFCV system can be performed either with the spreadsheet supplied (sc) or with your own tools (e.g., Lotus-123, Quattro). The data formats are designed to be as flexible as possible.

3.1.1 Using sc spreadsheet (Unix/VMS)

The sc spreadsheet used by XFCV on the Unix and VMS platforms is a modified version of the public domain distribution. It has been enhanced to facilitate its use by XFCV. The man page for the modified version of sc is included as an appendix to this document. Refer to it for more information on using the spreadsheet.

3.1.2 Using other spreadsheets

Other spreadsheet programs (or even a text editor) may be used to edit data for the system. Note, however, that if you use something other than the sc spreadsheet included, you will have to make sure that the data is in the proper format. Generally, most commercial spreadsheets have a mechanism for writing out data in a text format. Follow the instructions for doing this with the spreadsheet you are using.

3.2 Data Scaling

You may wish to scale the raw data before analysis by the XFCV system. The *Scale Data* component allows you to perform the desired data scaling.

Typically, you will want to auto-scale your data, so as to remove the influence of the difference in magnitude between descriptors.

3.3 Data Analysis

3.3.1 Principal Coordinate plotting

Performing the PC plot is typically the first step in the analysis you will perform. Once you have computed the PC map, you are advised to view the resulting coordinates, in order to verify if you have performed the correct data scaling, before performing the FCV cluster analysis (which is typically more time-consuming that the PC plotting function).

3.3.2 FCV Clustering

The FCV clustering analysis is the second step in the analysis. You should obtain from the PC plot done in the first pass a rough idea of what the geometric cluster centers are for your data set. You will need to input these for the clustering algorithm to function correctly.

3.4 Loading data sets

Once you have performed the editing, scaling, and analysis steps on the data set, you are ready to load the complete data set. You need to load the PC plot data first (dataset.pcdata). You may then optionally load the class information (dataset.class) and the sample ID's (which contain the known class information for the data set, if any).

3.5 Visualizing data sets

There are several ways you can visualize your data set, depending on the information you wish to obtain from it. The simplest is a simple PC plot, with no class information or color, as is shown in Figure 3-1. From there, the methods increase the amount of information displayed.

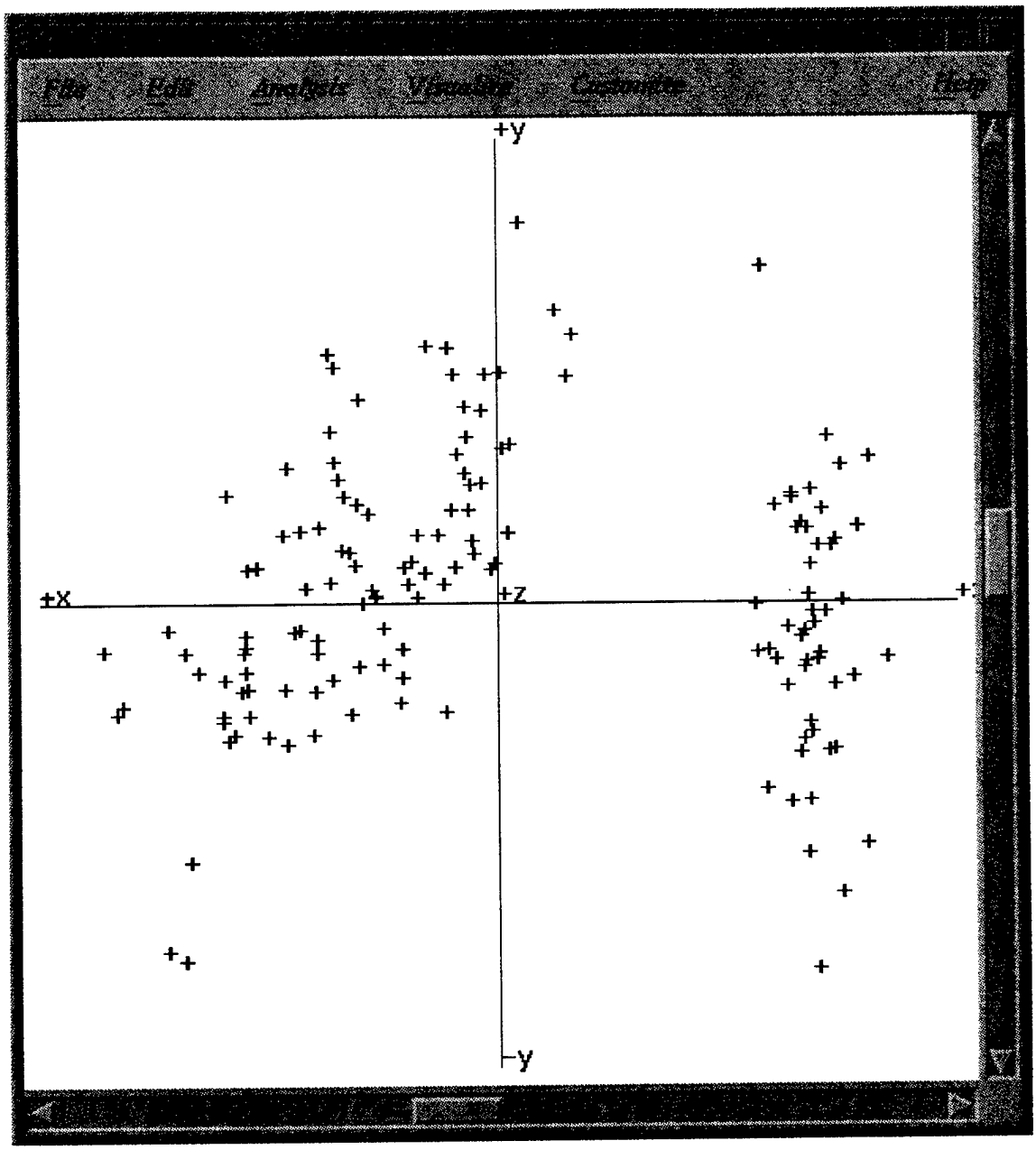


Figure 3-1 Iris Data Set with No Color and User Defined Markers

3.5.1 Static color visualization

The next basic visualization technique is static coloring. With this method, you wish to see class information, either strictly by the largest class membership value of each sample (as is shown in Figure 3-2), or by the define class value (as is shown in Figure 3-3). With each of these techniques, you are viewing each sample in terms of the base color defined for the particular class of interest (eg, red, green, blue, etc).

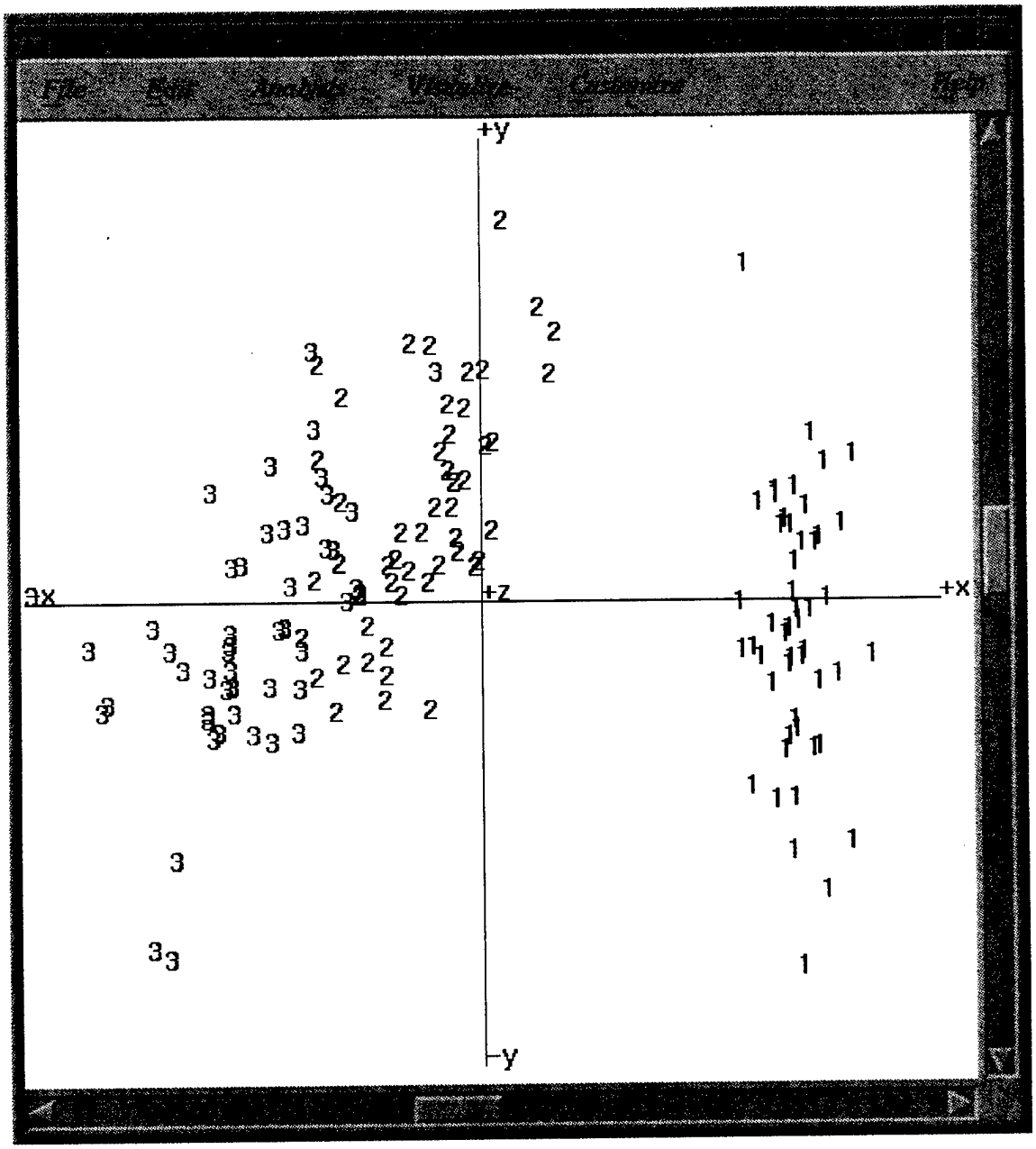


Figure 3-2 Iris Data Set with Static Coloring by Largest Class Membership

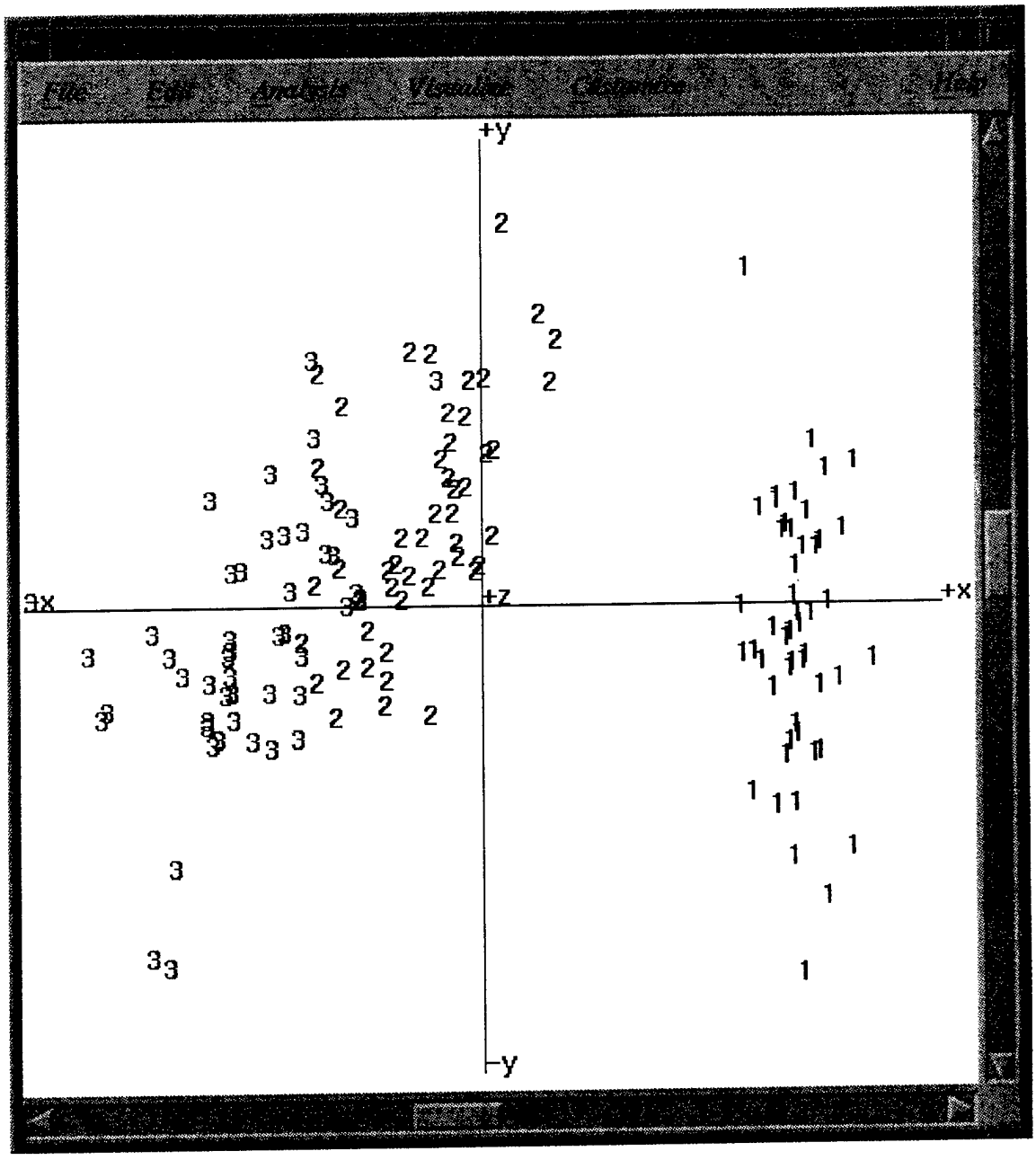


Figure 3-3 Iris Data Set with Static Coloring by Defined Class Membership

3.5.2 Fuzzy color visualization

Fuzzy color visualization is a variant of the various static coloring techniques. What fuzzy coloring allows you to see are the samples which are 'close' in membership to another class. The amount of 'closeness' or fuzz, is user-defineable. By default, a value of 0.05 is used by XFCV. When viewing a data set by fuzzy coloring, you will see any samples which are on the 'edge' between two classes, as is shown in Figure 3-4. Fuzzy samples are displayed as a combination of the two classes in which they have the highest membership values, in the form 'n/m'. In Figure 3-5, the fuzzy color tolerance is increased from 0.05 to 0.10. You will note that more values show up as 'border' values, mostly between class 2 and 3.

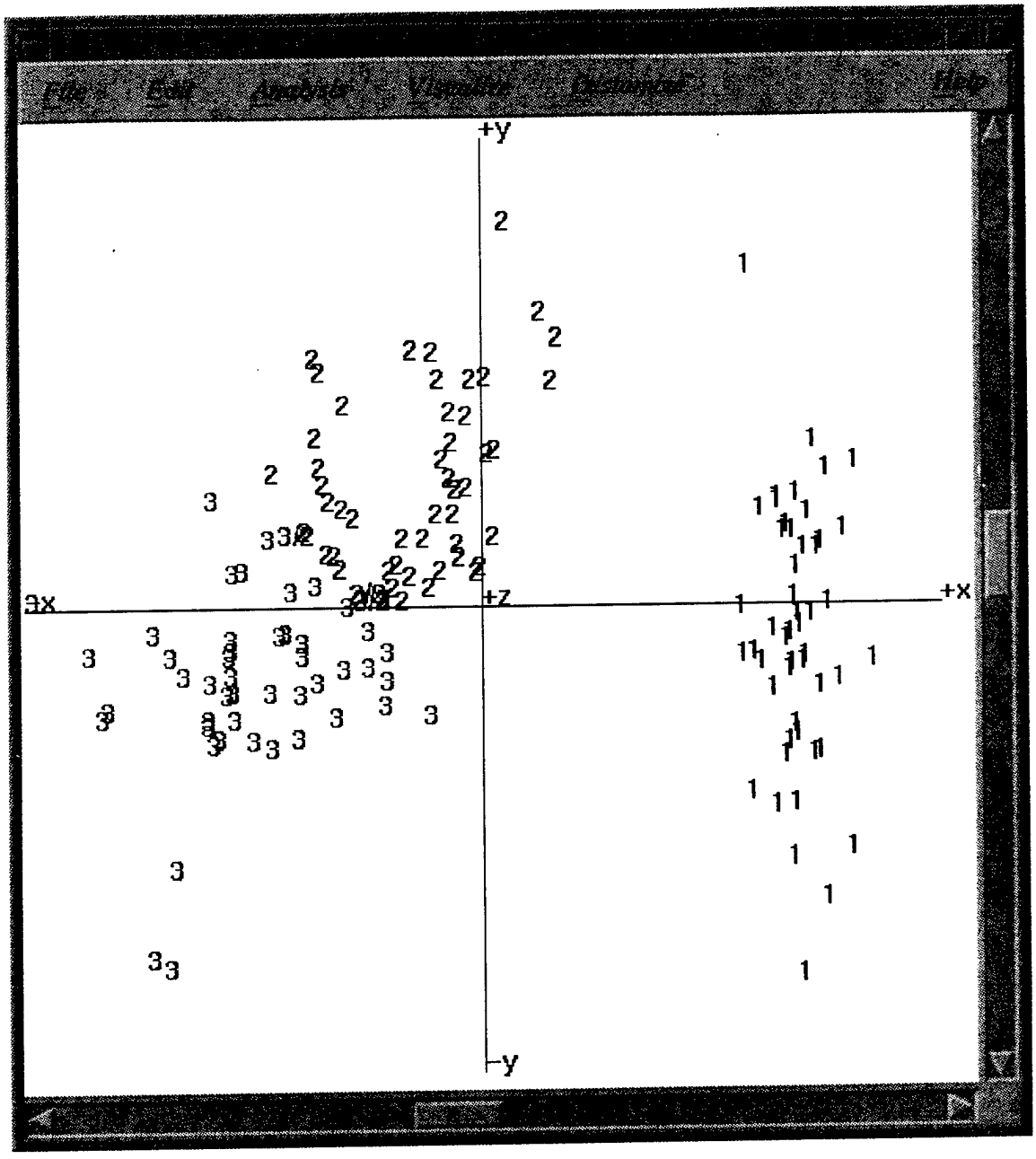


Figure 3-4 Iris Data Set with Fuzzy Coloring (Fuzz tolerance = 0.05)

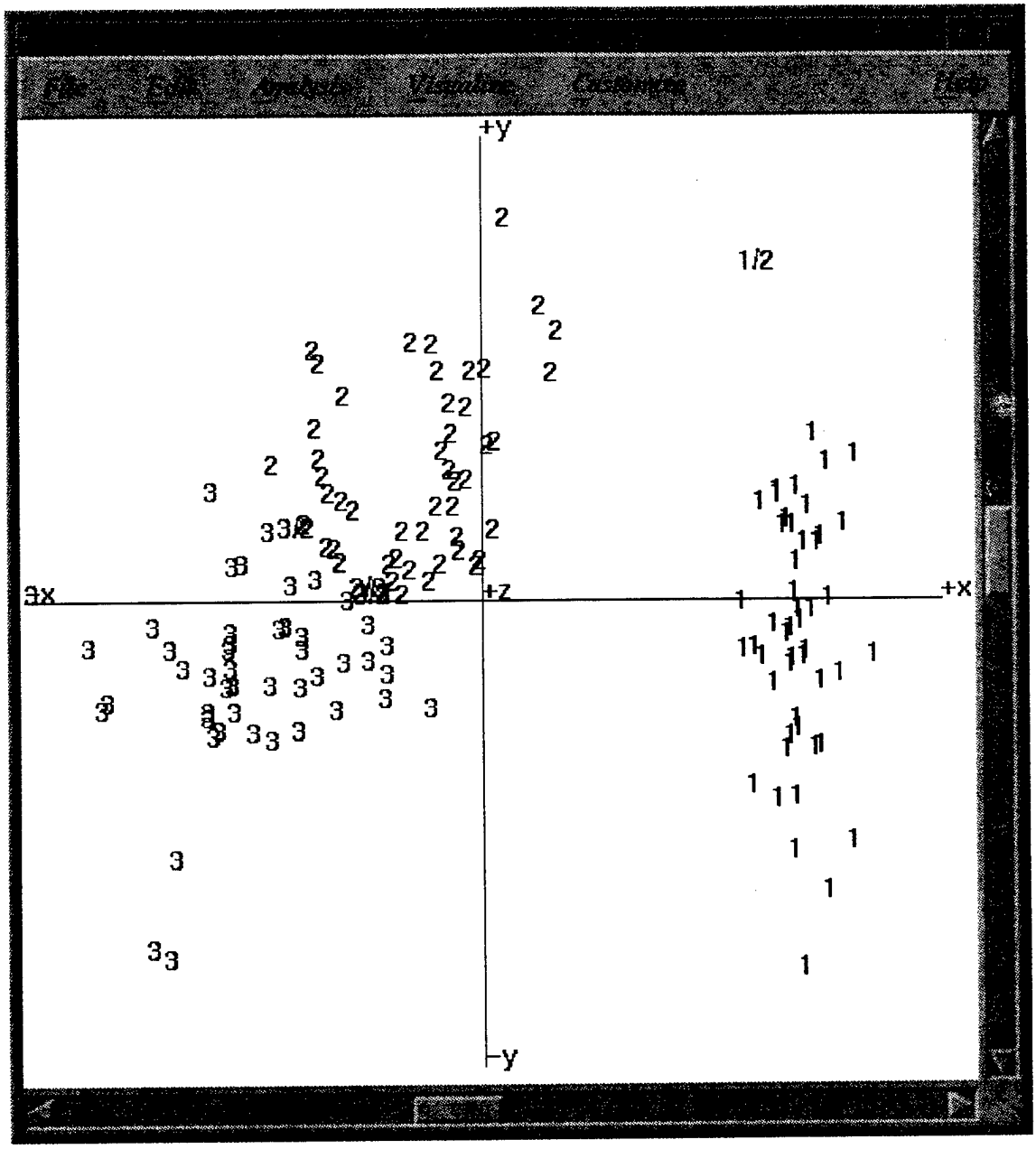


Figure 3-5 Iris Data Set with Fuzzy Coloring (Fuzz tolerance = 0.10)

3.5.3 Dynamic color visualization

Dynamic color visualization provides you with a means of 'seeing' the class membership values of each sample for all classes being viewed. The color of each sample point is a combination of the base color of each cluster. The amount of color contributed by a class is dependent on the amount of membership which that sample holds in that class. You also view the dynamic coloring as the amount of class membership which a sample has in its primary class and leave out other classes (intra-cluster coloring). Figure 3-6 shows an example of dynamic coloring, with inter-cluster coloring being used. Note the 'smear' of color near the edges of each cluster. Points which are further away from the cluster center will tend to have less of the color from the primary class and more of a class they are nearer to.

In addition to viewing the clusters as groups of points, you can elect to see the 'size' of the cluster by displaying it as a starburst. In starburst mode, a ray of the color of the sample is drawn from the cluster center to each sample in the cluster. As shown in Figure 3-7, this gives an idea of the overall volume of the cluster.

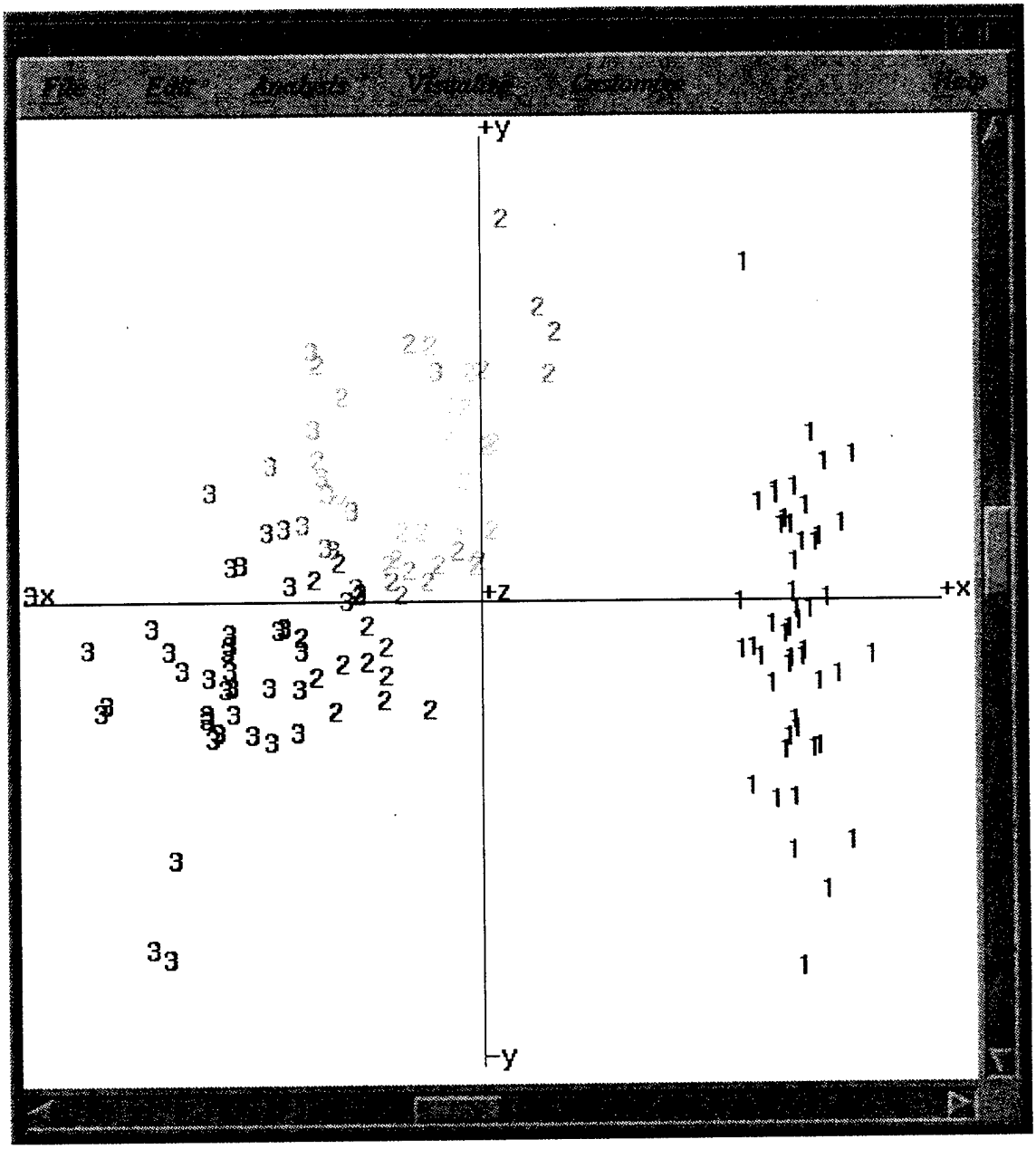


Figure 3-6 Iris Data Set with Dynamic Coloring (Inter-Cluster)

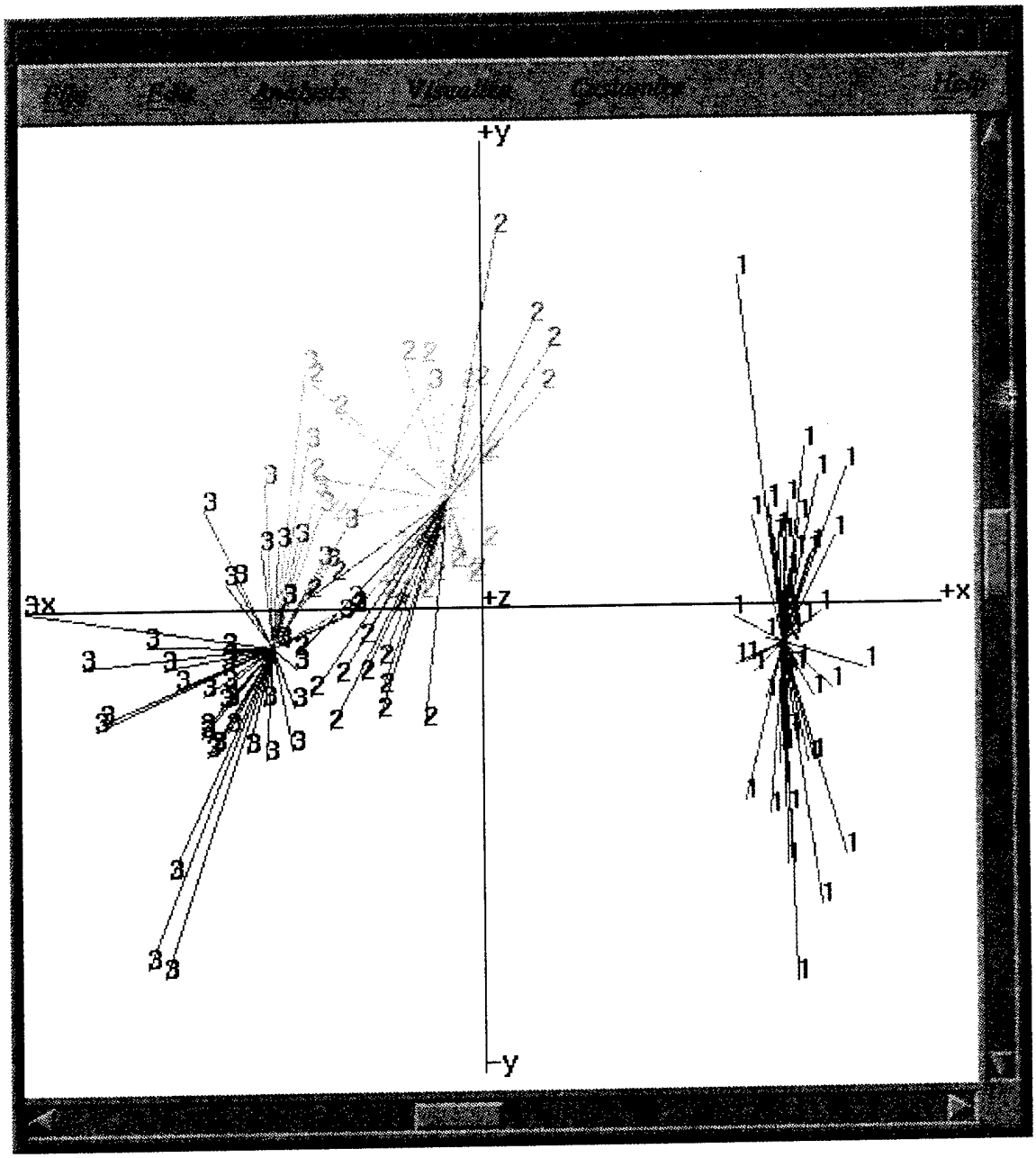


Figure 3-7 Iris Data Set with Dynamic Coloring viewed as Starburst

3.5.4 Modifying visualization parameters

Finally, there a numerous way in which you can modify the behaviour of a particular visualization mode, in order to achieve particular effects. The *Customize Visualization Parameters* option allows you to modify such things as the base color of each cluster, the markers for each cluster, the fuzzy color tolerance, and the dynamic coloring mode (inter-cluster vs. intra-cluster). By changing the cluster coloring, you can accentuate the similarities or differences between clusters, in order to get a better idea of the nature of the data.

REFERENCES

1. Jurs, P.C., Lavine, B.K. Stouch, T.R., "Pattern Recognition Studies of Complex Chromatographic Data Sets," NBS Journal of Research, December, 1985, pp. 543-549.

2. Belcher, A.M., Smith A.B., Jurs, P.C., Lavine, B.K., Epple, G. "Analysis of Chemical Signals in a Primate Species (Saguinus fusciocollis): Use of Behavior, Chemical, and Pattern Recognition Methods," Journal of Chemical Ecology, May, 1986, pp. 513-522.

3. Lavine, B.K., Carlson, D., "European Bee or Africanized Bee? Species Identification Through Chemical Analysis," Analytical Chemistry, March, 1987, 59, pp. 468A- 472A.

4. Dunn, W.J., Stalling, D.L., Schwartz, T.R., Hogan, J.W., Petty, J.D., "Pattern Recognition for Classification and Determination of Polychlorinated Biphenyls in Environmental Samples," Analytical Chemistry, September, 1984, 56, pp. 1308-1315.

5. Nilson, N.J. Learning Machines, McGraw-Hill, New York, 1965.

6. Lachenbruch, P.A. Discriminant Analysis, Hafner Press, New York, N.Y., 1975.

7. Dunn, W.J., Wold, S., Martin, Y.C., "Structure-Activity Study of Beta-Adrenegic Agents Using the SIMCA Method of Pattern Recognition," Journal of Medicinal Chemistry, May 1978, 21, pp. 922-932.

8. Lavine, B.K., Jurs, P.C., Henry, D.R., "Chance Classifications by Nonparametric Linear Discriminant Functions," Journal of Chemometrics, January 1988, 2, pp. 1-10.

9. Lavine, B.K., Henry, D.R., "Monte Carlo Studies of Nonparametric Linear Discriminant Functions," Journal of Chemometrics, January 1988, 2, pp. 85-90.

10. Bezdek, J.C., Coray, C.R., Gunderson, R.W., Watson, J.D., "Detection and Characterization of Cluster Substructure I. Linear Structure: Fuzzy c-Lines," SIAM Journal of Applied Mathematics, April 1981, 40, pp. 339-357.

11. Bezdek, J.C., Coray, C.R., Gunderson, R.W., Watson, J.D., "Detection and Characterization of Cluster Substructure II," SIAM Journal of Applied Mathematics, April 1981, 40, pp. 358-372.

12. Jolliffe, I. P., Principal Component Analysis, Springer-Verlag Press, New York, 1986.

13. Leu, C.H., Bowies, D.S., Editors, Procedures of the Vancouver Symposium, August 1987, IAHS Publication No.166., pp. 56-70.

14. Gunderson, R.W., Editor, 5th Annual Nordic Conference on Applied Statistics, Stockkand Foriag Publishers, Stravanger, Norway, 1985, pp. 65-85.

15. Lavine, B.K., Morel, L., Vander Meer, R.K., Gunderson, R.W., Han, J.H., Bonanno, A., Stine, A., "Pattern Recognition Studies in Chemical Communication:Nestmate Recognition in Camponotus floridanus," Chemometrics and Intelligent Laboratory Instrumentation, September 1990, 9, pp. 107-114.

16. Ekftrom, M.P. Digital Imaging Techniques, Academic Press, Orlando, Florida, 1984.

17. Hall, J.R, Glysson, G.D., Editors, Monitoring Water in the 1990's: Meeting New Challenges, ASTM STP 1102, American Society for Testing and Materials, Philadelphia, PA., 1991, pp. 578-597.

18. Mayfield, H.T., Bertsch, W., "Set-Up: A Program for Peak Matching," Journal of Computer Applications in the Laboratory, January 1983, 1, pp. 130-137.

19. Kowalski, Editor, Chemometrics: Theory and Application, American Chemical Society, 52, Washington, DC, 1977, pp. 243-280.

20. Stone, M.,"Cross-Validitory Choice and Assessment of Statistical Predictions," Journalof the Royal Statistical Society, February 1974, 36, pp. 111-121.

Appendix A
SC Spreadsheet *man* page

SC Spreadsheet *man* page

NAME

sc - spreadsheet calculator

SYNOPSIS

sc [-c] [-m] [-n] [-r] [-x] [*file*]

DESCRIPTION

The spreadsheet calculator *sc* is based on rectangular tables much like a financial spreadsheet. When invoked it presents you with a table organized as rows and columns of cells. If invoked without a *file* argument, the table is initially empty. Otherwise file is read in (see the *Get* command below). Each cell may have associated with it a numeric value, a label string, and/or an expression (formula) which evaluates to a numeric value or label string, often based on other cell values.

For a on-line tutorial, type the command:

sc /usr/local/lib/sc/tutorial.sc

To print a quick reference card, type the command:

scqref | [your_printer_commmand]

OPTIONS

-c Start the program with the recalculation being done in column order.

-m Start the program with automatic recalculation disabled. The spreadsheet will be recalculated only when the "@" command is used.

-n Start the program in quick numeric entry mode (see below).

-r Start the program with the recalculation being done in row order (default option).

-x Cause the *Get* and *Put* commands (see below) to encrypt and decrypt data files.

-R Start the program with automatic newline action set to increment the row (see below).

-C Start the program with automatic newline action set to increment the column (see below).

All of these options can be changed with the ^*T* and *S* commands (see below) while *sc* is running. Options specified when *sc* is invoked override options saved in the data file.

General Information

The screen is divided into four regions. The top line is for entering commands and displaying cell values. The second line is for messages from *sc*. The third line and the first four columns show the column and row numbers, from which are derived cell addresses, e.g. *A0* for the cell in column *A* row *0*. Note that column names are case-insensitive: you can enter *A0* or *a0*.

The rest of the screen forms a window looking at a portion of the table. The total umber of display rows and columns available, hence the number of table rows and columns displayed, is set by *curses(3)* and may be overridden by setting the INES and COLUMNS environment variables, respectively.

The screen has two cursors: a cell cursor, indicated by a highlighted cell and a "<" on the screen, and a character cursor, indicated by the terminal's hardware cursor. The cell and character cursors are often the same. They differ when you type a command on the top line.

If a cell's numeric value is wider than the column width (see the *f* command), the cell is filled with asterisks. If a cell's label string is wider than the column width, it is truncated at the start of the next non-blank cell in the row, if any. Cursor control commands and row and column commands can be prefixed by a numeric argument which indicates how many times the command is to be executed. You can type ^*U* before a repeat count if quick numeric entry mode is enabled or if the number is to be entered while the character cursor is on the top line.

Commands which use the terminal's control key, such as ^*N*, work both when a command is being typed and when in normal mode.

Changing Options

^*To* Toggle options. This command allows you to switch the state of one option selected by *o*. A small menu lists the choices for *o* when you type ^*T*. The options selected are saved when the data and formulas are saved so that you will have the same setup next time you enter the spreadsheet.

a Automatic Recalculation. When set, each change in the spreadsheet causes the entire spreadsheet to be recalculated. Normally this is not noticeable, but for very large spreadsheets, it may be faster to clear automatic recalculation mode and update the spreadsheet via explicit "@" commands. Default is automatic recalculation on.

c Current cell highlighting. If enabled, the current cell is highlighted (using the terminal's standout mode, if available) in addition to being marked by the cell cursor.

e External function execution. When disabled, external functions (see *@ext()* below) are not called. This saves a lot of time at each screen update. External functions are disabled by default. If disabled, and external functions are used anywhere, a warning is printed each time the screen is updated, and the result of *@ext()* is the value from the previous call, if any, or a null string.

l Autolabeling. If enabled, using the define command (*/d*) causes a label to be automatically generated in the cell to the left of the defined cell. This is only done if the cell to the left is empty. Default is enabled.

n Quick numeric entry. If enabled, a typed digit is assumed to be the start of a numeric value for the current cell, not a repeat count, unless preceded by ^*U*. The cursor controls (^*P*, ^*N*, ^*B*, ^*F*) in this mode will end a numeric entry.

t Top line display. If enabled, the name and value of the current cell is displayed on the top line. If there is an associated label string, the first character of the string value is "|" for a centered string, "<" for a leftstring or ">" for a rightstring (see below), followed by "*string*" for a constant string or {*expr*} for a string expression. A constant string may be preceeded with a backslash ('\'). In this case the constant string will be used as a "wheel" to fill a column, e.g. "\-" for a line in a column, and "\Yeh " for "Yeh Yeh Ye". If the cell has a numeric value, it

follows as [*value*], which may be a constant or expression.

x Encryption. See the *-x* option.

$ Dollar prescale. If enabled, all numeric constants (not expressions) which you enter are multipled by 0.01 so you don't have to keep typing the decimal point if you enter lots of dollar figures.

r Newline action. This option toggles between three cases. The default is no action. If this option is used once, after each command which is terminated by a newline character is completed, the current cell will be moved down one row. If this option is used again, after each command which is terminated by a newline character is completed, the current cell will be moved right one column. Another use of this option will restore the default action.

z Set newline action limits. This option sets limits to the newline action option above. When this option is invoked, the row and column of the current cell are remembered. If a later newline action would take the current cell to the right of the remembered column, then the current cell is instead moved to the first column of the next row. If a newline action would take the current cell below the remembered row, then the current cell is instead moved to the top row of the next column.

The quick numeric entry, newline action and set newline action limits options can be combined to allow very quick entry of large amounts of data. If all the data to be entered is in a single row or column then setting the quick numeric entry and the appropriate newline action will allow the numbers to be entered without any explicit commands to position the current cell or enter a number.

If the data entry involves several entries in each row for many rows, then setting the quick numeric entry option, setting the newline action to move right after each entry and setting the newline action limits on the last column on which data should be entered will allow the data to entered quickly. If necessary, columns which do not need data to be entered can be hidden with the *z* command. Similar arrangements can be made for entering several rows of data in each column.

S Set options. This command allows you to set various options. A small menu lists the options that cannot be changed through *^T* above.

byrows / bycols
Specify the order cell evaluation when updating. These options also affect the order in which cells are filled (see */f*) and whether a row or column is cleared by an *x* command.

iterations=n
Set the maximum number of recalculations before the screen is displayed again. *Iterations* is set to 10 by default.

tblstyle=s
Control the output of the *T* command. *s* can be: *fcv* (default) to give comma delimited fields, *0* to give colon delimited fields, with no tbl control lines; *tbl* to give colon delimited fields, with tbl(1) control lines; *latex* to give a LaTeX tabular environment; *slatex* to give a *SLaTeX* (Scandinavian LaTeX) tabular environment; and *tex* to give a TeX simple tabbed alignment with ampersands as delimiters.

Other *Set* options are normally used only in *sc* data files since they are available through ^*T*. You can also use them interactively.

autocalc / !autocalc
Set/clear auto recalculation mode.

numeric / !numeric
Set/clear numeric mode.

prescale / !prescale
Set/clear numeric prescale mode.

extfun / !extfun
Enable/disable external functions.

cellcur / !cellcur
Set/clear current cell highlighting mode.

toprow / !toprow
Set/clear top row display mode.

rndinfinity / !rndinfinity
default: round-to-even (banker's round), *.5 will round to the closest even number; doing a 'set rndinfinity' will round *.5 up to the next integer (rounding to infinity).

craction=n

Set the newline action. n can be: 0 (default) to give no action; 1 to move down after each entry; or 2 to move right after each entry.

rowlimit=n

Set the remembered limit for the maximum row below which the current cell will be moved to the top of the next column if the newline action is set to move the current cell down. *n* can be -1 (default) to disable this facility.

collimit=n

Set the remembered limit for the maximum column to the right of which the current cell will be moved to the left of the next row if the newline action is set to move the current cell right. *n* can be -1 (default) to disable this facility.

Cursor Control Commands

^P Move the cell cursor up to the previous row.

^N Move the cell cursor down to the next row.

^B Move the cell cursor backward one column.

^F Move the cell cursor forward one column.

h, j, k, l
If the character cursor is not on the top line, these are alternate, *vi*-compatible cell cursor controls (left, down, up, right). Space is just like *l* (right).

H, J, K, L
If the character cursor is not on the top line, these move the cursor by half pages (left, down, up, right).

^H If the character cursor is not on the top line, *^H* is the same as *^B*.

SPACE
If the character cursor is not on the top line, the space bar is the same as *^F*.

TAB
If the character cursor is on the top line, TAB starts a range (see below). Otherwise, it is the same as *^F*.

Arrow Keys
The terminal's arrow keys provide another alternate set of cell cursor controls if they exist and are supported in the appropriate termcap entry. Some terminals have arrow keys which conflict with other control key codes. For example, a terminal might send *^H* when the back arrow key is pressed. In these cases, the conflicting arrow key performs the same function as the key combination it mimics.

^ Move the cell cursor up to row 0 of the current column.

Move the cell cursor down to the last valid row of the current column.

0 Move the cell cursor backward to column A of the current row. This command must be prefixed with *^U* if quick numeric entry mode is enabled.

$ Move the cell cursor forward to the last valid column of the current row.

b Scan the cursor backward (left and up) to the previous valid cell.

w Scan the cursor forward (right and down) to the next valid cell.

^Ed

Go to end of range. Follow *^E* by a direction indicator such as *^P* or *j*. If the cell cursor starts on a non-blank cell, it goes in the indicated direction until the last non-blank adjacent cell. If the cell cursor starts on a blank cell, it goes in the indicated direction until the first non-blank cell. This command is useful when specifying ranges of adjacent cells (see below), especially when the range is bigger than the visible window.

g Go to a cell. *sc* prompts for a cell's name, a regular expression surrounded by quotes, or a number. If a cell's name such as *ae122* or a the name of a defined range is given, the cell cursor goes directly to that cell. If a quoted regular expression such as " Tax Table " or " ^Jan[0-9]*$ " is given, *sc* searches for a cell containing a string matching the regular expression. See *regex(3)* or *ed(1)* for more details on the form of regular expressions. If a number is given, sc will search for a cell containing that number. Searches for either strings or numbers proceed forward from the current cell, wrapping back to a0 at the end of the table, and terminate at the current cell if the string or number is not found. You may also go to a cell with an ERROR (divide by zero, etc in this cell) or INVALID (references a cell containing an ERROR). *gerror* will take you to the next ERROR, while *ginvalid* take you to the next invalid. The last *g* command is saved, and can be re-issued by entering *g<return>*.

Cell Entry and Editing Commands

Cells can contain both a numeric value and a string value. Either value can be the result of an expression, but not both at once, i.e. each cell can have only one expression associated with it. Entering a valid numeric expression alters the cell's previous numeric value, if any, and replaces the cell's previous string expression, if any, leaving only the previously computed constant label string. Likewise, entering a valid string expression alters the cell's the previous label string, if any, and replaces the cell's previous numeric expression, if any, leaving only the previously computed constant numeric value.

= Enter a numeric constant or expression into the current cell. *sc* prompts for the expression on the top line. The usual way to enter a number into a cell is to type =, then enter the number in response to the prompt on the top line. The quick numeric entry option, enabled through the -n option or ^*T* command, shows the prompt when you enter the first digit of a number (you can skip typing =).

< Enter a label string into the current cell to be flushed left against the left edge of the cell.

" Enter a label string into the current cell to be centered in the column.

> Enter a label string into the current cell to be flushed right against the right edge of the cell.

F Enter a format string into the current cell. This format string overrides the precision specified with the *f* command. The format only applies to numeric values. The following characters can be used to build a format string:

Digit placeholder. If the number has fewer digits on either side of the decimal point than there are # characters in the format, the extra # characters are ignored. The number is rounded to the number of digit placeholders as there are to the right of the decimal point. If there are more digits in the number than there are digit placeholders on the left side of the decimal point, then those digits are displayed.

0 Digit placeholder. Same as for # except that the number is padded with zeroes on either side of the decimal point. The number of zeroes used in padding is determined by the number of digit placeholders after the *0* for digits on the left side of the decimal point and by the number of digit placeholders before the *0* for digits on the right side of the decimal point.

. Decimal point. Determines how many digits are placed on the right and left sides of the decimal point in the number. Note that numbers smaller than 1 will begin with a decimal point if the left side of the decimal point contains only a # digit placeholder. Use a *0* placeholder to get a leading zero in decimal formats.

% Percentage. For each % character in the format, the actual number gets multiplied by 100 (only for purposes of formatting -- the original number is left unmodified) and the % character is placed in the same position as it is in the format.

, Thousands separator. The presence of a ',' in the format (multiple commas are treated as one) will cause the number to be formatted with a ',' separating each set of three digits in the integer part of the number with numbering beginning from the right end of the integer.

\ Quote. This character causes the next character to be inserted into the formatted string directly with no special interpretation.

E- E+ e- e+
Scientific format. Causes the number to formatted in scientific notation. The case of the *E* or *e* given is preserved. If the format uses a +, then the sign is always given for the exponent value. If the format uses a -, then the sign is only given when the exponent value is negative. Note that if there is no digit placeholder following the + or -, then that part of the formatted number is left out. In general, there should be one or more digit placeholders after the + or -.

; Format selector. Use this character to separate the format into two distinct formats. The format to the left of the ; character will be used if the number given is zero or positive. The format to the right of the ; character is used if the number given is negative.

Some example formats are integer (*0* or #), fixed (*0.00*), percentage (*0%* or *0.00%*), scientific (*0.00E+00*), and currency (*$#,0.00;($#,0.00)*).

Strings you enter must start with ". You can leave off the trailing " and *sc* will add it for you. You can also enter a string expression by backspacing over the opening " in the prompt.

e Edit the value associated with the current cell. This is identical to

= except that the command line starts out containing the old numeric value or expression associated with the cell. The editing in this mode is *vi*-like.

^*H* Move back a character.

\+ Forward through history (neat) (same as *j*).

\- Backward through history (neat) (same as *k*)

ESC
Done editing

TAB
Mark && append a range (ex: A0:A0) *TAB*, move around within a range; *TAB*, append range string.

CR Save

$ Goto last column

. Insert current dot buffer

/ Search for a string in the history

ESC	edit the string you typed
CR	search
^*H*	backspace

0 Goto column *0*

D Delete to send

I Insert at column 0; *ESC* revert back to edit mode

R Replace mode; *ESC* revert back to edit mode

X Delete the char to the left

a Append after cursor; *ESC* revert back to edit mode

b Move back a word

c Change mode; ESC revert back to edit mode

d Delete ...

b back word
f forward (right)
h back char
l forward
t delete forward up to a given char (next char typed)
w delete next word forward

f Find the next char typed

h Move left a char

i Insert before cursor; ESC revert back to edit mode

j Forward through history (neat) (same as +)

k Backward through history (neat) (same as -)

l Move right a char

n Continue search

q Stop editing

r Replace char

t Goto a char

u Undo

w Forward a word

x Delete the current char (moving to the right)

E Edit the string associated with the current cell. This is identical to <, ", or > except that the command line starts out containing the old string value or expression associated with the cell. SEE *e* ABOVE.

To enter and edit a cell's number part, use the = and *e* commands. To enter and edit a cell's string part, use the <, ", >, and *E* commands. See the sections below on numeric and string expressions for more information.

x Clear the current cell. Deletes the numeric value, label string, and/or numeric or string expression. You can prefix this command with a count of the number of cells on the current row to clear. The current column is used if column recalculation order is set. Cells cleared with this command may be recalled with any of the *pull* commands (see below).

m Mark a cell to be used as the source for the *copy* command.

c Copy the last marked cell to the current cell, updating row and column references in its numeric or string expression, if any.

\+ If not in numeric mode, add the current numeric argument (default 1) to the value of the current cell. In numeric mode, + introduces a new numeric expression or value, the same as =.

\- If not in numeric mode, subtract the current numeric argument (default 1) from the value of the current cell. In numeric mode, - introduces a new, negative, numeric expression or value, like =.

RETURN

If you are not editing a cell (top line is empty), pressing *RETURN* will make *sc* enter insert mode. At this point you may type any valid command or press *ESC* once to edit.

File Commands

G Get a new database from a file. If encryption is enabled, the file is decrypted before it is loaded into the spreadsheet.

P Put the current database into a file. If encryption is enabled, the file is encrypted before it is saved.

W Write a listing of the current database into a file in a form that matches its appearance on the screen. This differs from the *Put* command in that its files are intended to be reloaded with *Get*, while *Write* produces a file for people to look at. Hidden rows or columns are not shown when the data is printed.

T Write a listing of the current database to a file, but include delimiters suitable for processing by the *tbl*, *LaTeX*, or *TeX* table processors. The delimiters are controlled by the *tblstyle* option. See *Set* above. The delimiters are are a colon (:) for style *0* or *tbl* and an ampersand (&) for style *latex* or *tex*.

With the *Put*, *Write*, and *Table* commands, the optional range argument writes a subset of the spreadsheet to the output file.

With the *Write* and *Table* commands, if you try to write to the last file used with the *Get* or *Put* commands, or the file specified on the command line when *sc* was invoked, you are asked to confirm that the (potentially) dangerous operation is really what you want.

The three output commands, *Put*, *Write*, and *Table*, can pipe their (unencrypted only) output to a program. To use this feature, enter | *program* to the prompt asking for a filename. For example, to redirect the output of the Write command to the printer, you might enter | *lpr -p*.

M Merge the database from the named file into the current database. Values and expressions defined in the named file are read into the current spreadsheet overwriting the existing entries at matching cell locations.

R Run macros. Since *sc* files are saved as ASCII files, it is possible to use them as primitive macro definition files. The *Run* command makes this easier. It's like the *Merge* command, but prints a saved path name as the start of the filename to merge in. The string to use is set with the *Define* command. To write macros, you must be familiar with the file format written by the *Put* command. This facility is still primitive and could be much improved.

D Define a path for the *Run* command to use.

All file operations take a filename as the first argument to the prompt on the top line. The prompt supplies a " to aid in typing in the filename. The filename can also be obtained from a cell's label string or string expression. In this case, delete the leading " with the backspace key and enter a cell name such as *a22* instead. If the resulting string starts with |, the rest of the string is interpreted as a UNIX command, as above.

Row and Column Commands

These commands can be used on either rows or columns. The second letter of the command is either a row designator (one of the characters *r*, ^*B*, ^*F*, *h*, *l*) or a column designator (one of *c*, ^*P*, ^*N*, *k*, *j*). A small menu lists the choices for the second letter when you type the first letter of one of these commands. Commands which move or copy cells also modify the row and column references in affected cell expressions. The references may be frozen by using the *fixed* operator or using the *$* character in the reference to the cell (see below).

ir, ic

Insert a new row (column) by moving the row (column) containing the cell cursor, and all following rows (columns), down (right) one row (column). The new row (column) is empty.

ar, ac

Append a new row (column) immediately following the current row (column). It is initialized as a copy of the current one.

dr, dc

Delete the current row (column).

pr, pc, pm

Pull deleted rows (columns) back into the spreadsheet. The last deleted set of cells is put back into the spreadsheet at the current location. *pr* inserts enough rows to hold the data. *pc* inserts enough columns to hold the data. *pm* (merge) does not insert rows or columns; it overwrites the cells beginning at the current cell cursor location.

vr, vc

Remove expressions from the affected rows (columns), leaving only the values which were in the cells before the command was executed.

zr, zc

Hide (*zap*) the current row (column). This keeps a row (column) from being displayed but keeps it in the data base. The status of the rows and columns is saved with the data base so hidden rows and columns will be still be hidden when you reload the spreadsheet. Hidden rows or columns are not printed by the *W* command.

sr, sc

Show hidden rows (columns). Enter a range of rows (columns) to be revealed. The default is the first range of rows (columns) currently hidden. This command ignores the repeat count, if any.

f Set the output format to be used for printing the numeric values in each

cell in the current column. Enter three numbers: the total width in characters of the column, the number of digits to follow decimal points, and the format type. Format types are *0* for fixed point, *1* for scientific notation, *2* for engineering notation, and *3* for dates. Values are rounded off to the least significant digit displayed. The total column width affects displays of strings as well as numbers. A preceding count can be used to affect more than one column. This command has only a column version (no second letter).

@myrow, @mycol

Are functions that return the row or column of the current cell respectively. ex: The cell directly above a cell in the D column could then be accessed by *@nval("d",@myrow-1)*. NOTE: *@myrow* and *@mycol* can't be used in specifying ranges.

Range Commands

Range operations affect a rectangular region on the screen defined by the upper left and lower right cells in the region. All of the commands in this class start with a slash; the second letter of the command indicates which command. A small menu lists the choices for the second letter when you type /. *sc* prompts for needed parameters for each command. Phrases surrounded by square brackets in the prompt are informational only and may be erased with the backspace key.

Prompts requesting variable names may be satisfied with either an explicit variable name, such as *A10*, or with a variable name previously defined in a */d* command (see below). Range name prompts require either an explicit range such as *A10:B20*, or a range name previously defined with a */d* command. A default range shown in the second line is used if you omit the range from the command or press the TAB key (see below). The default range can be changed by moving the cell cursor via the control commands (*^P, ^N, ^B, ^F*) or the arrow keys. The cells in the default range are highlighted (using the terminal's standout mode, if available).

/x Clear a range. Cells cleared with this command may be recalled with any of the *pull* commands.

/v Values only. This command removes the expressions from a range of cells, leaving just the values of the expressions.

/c Copy a source range to a destination range. The source and destination maybe different sizes. The result is always one or more full copies of the source. Copying a row to a row yields a row. Copying a column to a column yields a column. Copying a range to anything yields a range. Copying a row to a column or a column to a row yields a range with as many copies of the source as there are cells in the destination. This command can be used to duplicate a cell through an arbitrary range by making the source a single cell range such as *b20:b20*.

/f Fill a range with constant values starting with a given value and increasing by a given increment. Each row is filled before moving on to the next row if row order recalculation is set. Column order fills each column in the range before moving on to the next column. The start and increment numbers may be positive or negative. To fill all cells with the same value, give an increment of zero.

/d Use this command to assign a symbolic name to a single cell or a rectangular range of cells on the screen. The parameters are the name, surrounded by "", and either a single cell name such as *A10* or a range such as *a10:b20*. Names defined in this fashion are used by the program

in future prompts, may be entered in response to prompts requesting a cell or range name, and are saved when the spreadsheet is saved with the *Put* command. Names defined must be more than two alpha characters long to differentiate them from a column names, and must not have embedded special characters. Names may include the character "_" or numerals as long as they occur after the first three alpha characters.

/l Use this command to lock the current cell or a range of cells, i.e make them immune to any type of editing. A locked cell can't be changed in anyway until it is unlocked.

/U This command is the opposite of the */l* command and thus unlocks a locked cell and makes it editable.

/s This command lists (shows) the currently defined range names. If there are no defined range names, then a message is given, otherwise it pipes output to *sort*, then to less. If the environment variable PAGER is set, its value is used in place of *less*.

/u Use this command to undefine a previously defined range name.

/F Use this command to assign a value format string (see the *F* cell entry command) to a range of cells.

Miscellaneous Commands

Q
q
^*C* Exit from *sc*. If you made any changes since the last *Get* or *Put*, *sc* asks about saving your data before exiting.

^*G*
ESC Abort entry of the current command.

? Enter an interactive help facility. Lets you look up brief summaries of the main features of the program. The help facility is structured like this manual page so it is easy to find more information on a particular topic.

! Shell escape. *sc* prompts for a shell command to run. End the command line with the RETURN key. If the environment variable **SHELL** is defined, that shell is run. If not, **/bin/sh** is used. Giving a null command line starts the shell in interactive mode. A second *!* repeats the previous command.

^*L* Redraw the screen.

^*R* Redraw the screen with special highlighting of cells to be filled in. This is useful for finding values you need to provide or update in a form with which you aren't familiar or of which you have forgotten the details.

It's also useful for checking a form you are creating. All cells which contain constant numeric values (not the result of a numeric expression) are highlighted temporarily, until the next screen change, however minor. To avoid ambiguity, the current range (if any) and current cell are not highlighted.

^*X* This command is similar to ^*R*, but highlights cells which have expressions. It also displays the expressions in the highlighted cells as left-flushed strings, instead of the numeric values and/or label strings of those cells. This command makes it easier to check expressions, at least when they fit in their cells or the following cell(s) are blank so the expressions can slop over (like label strings). In the latter case, the slop over is not cleared on the next screen update, so you may want to type ^*L* after the ^*X* in order to clean up the screen.

@ Recalculates the spreadsheet.

^*V* Type, in the command line, the name of the current cell (the one at the cell cursor). This is useful when entering expressions which refer to

other cells in the table.

^*W* Type, in the command line, the expression attached to the current cell. If there is none, the result is *?*.

^*A* Type, in the command line, the numeric value of the current cell, if any.

The ^*V*, ^*W*, and ^*A* commands only work when the character cursor is on the command line and beyond the first character.

TAB When the character cursor is on the top line, defines a range of cells via the cursor control commands or the arrow keys. The range is highlighted, starts at the cell where you typed *TAB*, and continues through the current cell cursor. Pressing *TAB* again causes the highlighted range to be entered into the command line and the highlighting to be turned off. This is most useful for defining ranges to functions such as *@sum()*. Pressing *)* acts just like typing the *TAB* key the second time and adds the closing *)*. Note that when you give a range command, you don't need to press the first *TAB* to begin defining a range starting with the current cell.

Variable Names

Normally, a variable name is just the name of a cell, such as *K20*. The value is the numeric or string value of the cell, according to context.

When a cell's expression (formula) is copied to another location via *copy* or *range-copy*, variable references are by default offset by the amount the formula moved. This allows the new formula to work on new data. If cell references are not to change, you can either use the *fixed* operator (see below), or one of the following variations on the cell name.

K20

References cell *K20*; the reference changes when the formula is copied.

K20

Always refers to cell *K20*; the reference stays fixed when the formula is copied.

$K20

Keeps the column fixed at column *K*; the row is free to vary.

K$20

Similarly, this fixes the row and allows the column to vary.

These conventions also hold on defined ranges. Range references vary when formulas containing them are copied. If the range is defined with fixed variable references, the references do not change.

fixed

To make a variable not change automatically when a cell moves, put the word *fixed* in front of the reference, for example: B1 * fixed C3.

Numeric Expressions

Numeric expressions used with the = and *e* commands have a fairly conventional syntax. Terms may be constants, variable names, parenthesized expressions, and negated terms. Ranges may be operated upon with range functions such as sum (*@sum()*) and average (*@avg()*). Terms may be combined using binary operators.

-e Negation.

e+e Addition.

e-e Subtraction.

*e*e* Multiplication.

e/e Division.

e1%e2
e1 mod e2.

e^e Exponentiation.

e<e
e<=e
e=e
e!=e
e>=e
e>e Relationals: true (1) if and only if the indicated relation holds, else false (0). Note that <=, *!=*, and >= are converted to their *~()* equivalents.

~e Boolean operator NOT.

e&e Boolean operator AND.

e|e Boolean operator OR.

@if(e,e,e)
e?e:e
Conditional: If the first expression is true then the value of the second is returned, otherwise the value of the third.

Operator precedence from highest to lowest is:

-, ~
^
*, /
+, -
<, <=, =, !=, >=, >
&
|
?:

Built-in Range Functions

These functions return numeric values.

@sum(r) Sum all valid (nonblank) entries in the region whose two corners are defined by the two variable names (e.g. *c5:e14*) or the range name specified.

@prod(r) Multiply together all valid (nonblank) entries in the specified region.

@avg(r) Average all valid (nonblank) entries in the specified region.

@count(r) Count all valid (nonblank) entries in the specified region.

@max(r) Return the maximum value in the specified region. See also the multi argument version of *@max* below.

@min(r) Return the minimum value in the specified region. See also the multi argument version of *@min* below.

@stddev(r) Return the sample standard deviation of the cells in the specified region.

@lookup(e,r)
@lookup(se,r)

Evaluates the expression then searches through the range *r* for a matching value. The range should be either a single row or a single column. The expression can be either a string expression or a numeric expression. If it is a numeric expression, the range is searched for the last value less than or equal to *e*. If the expression is a string expression, the string portions of the cells in the range are searched for an exact string match.

The value returned is the numeric value of the next row and the same column as the match, if the range was a single row, or the value from the next column and the same row as the match if the range was a single column.

@hlookup(e,r,n)
@hlookup(se,r,n)

Evaluates the expression then searches through the first row in the range *r* for a matching value. The expression can be either a string expression or a numeric expression. If it is a numeric expression, the row is searched for the the last value than or equal to *e*. If the expression is a string expression, the string portions of the cells in the row are searched for an exact string match.

The value returned is the numeric value from the same column *n* rows below the match.

@vlookup(e,r,n)
@vlookup(se,r,n)

Evaluates the expression then searches through the first column in the range *r* for a matching value. The expression can be either a string expression or a numeric expression. If it is a numeric expression, the column is searched for the last value less than or equal to *e*. If the expression is a string expression, the string portions of the cells in the column are searched for an exact string match.

The value returned is the numeric value from the same row *n* columns to the right of the match.

@index(e,r)

Use the value of the expression *e* to index into the range *r*. The numeric value at that position is returned. The value 1 selects the first item in the range, 2 selects the second item, etc. *R* should be either a single row or a single column.

@stindex(e,r)

Use the value of *e* to index into the range *r*. The string value at that position is returned. The value 1 selects the first item in the range, 2 selects the second item, etc. The range should be either a single row or a single column.

Built-in Numeric Functions

All of these functions operate on floating point numbers (doubles) and return numeric values. Most of them are standard system functions more fully described in math(3). The trig functions operate with angles in radians.

@sqrt(e) Return the square root of *e*.

@exp(e) Return the exponential function of *e*.

@ln(e) Return the natural logarithm of *e*.

@log(e) Return the base 10 logarithm of *e*.

@floor(e) Return the largest integer not greater than *e*.

@ceil(e) Return the smallest integer not less than *e*.

@rnd(e) Round e to the nearest integer. default: round-to-even (banker's round), *.5 will round to the closest even number; *'set rndinfinity'* will round *.5 up to the next integer.

@round(e,n)
Round *e* to *n* decimal places. n may be positive to round off the right side of the decimal, and negative to round off the left side. See *@rnd(e)* above for rounding types.

@abs(e)
@fabs(e) Return the absolute value of *e*.

@pow(e1,e2)
Return *e1* raised to the power of *e2*.

@hypot(e1,e2)
Return *sqrt(e1*e1+e2*e2)*, taking precautions against unwarranted overflows.

pi @pi A constant quite close to *pi*.

@dtr(e) Convert *e* in degrees to radians.

@rtd(e) Convert *e* in radians to degrees.

@sin(e)

@cos(e)
@tan(e) Return trigonometric functions of radian arguments. The magnitude of the arguments are not checked to assure meaningful results.

@asin(e) Return the arc sine of *e* in the range -pi/2 to pi/2.

@acos(e) Return the arc cosine of *e* in the range 0 to pi.

@atan(e) Return the arc tangent of *e* in the range -pi/2 to pi/2.

@atan2(e1,e2)
Returns the arc tangent of *e1/e2* in the range -pi to pi.

@max(e1,e2,...)
Return the maximum of the values of the expressions. Two or more expressions may be specified. See also the range version of *@max* above.

@min(e1,e2,...)
Return the minimum of the values of the expressions. Two or more expressions may be specified. See also the range version of *@min* above.

@ston(se) Convert string expression *se* to a numeric value.

@eqs(se1,se2)
Return 1 if string expression *se1* has the same value as string expression *se2*, 0 otherwise.

@nval(se,e) Return the numeric value of a cell selected by name. String expression *se* must evaluate to a column name (*A-AE*) and *e* must evaluate to a row number (0-199). If *se* or *e* is out of bounds, or the cell has no numeric value, the result is 0. You can use this for simple table lookups. Be sure the table doesn't move unexpectedly! See also *@sval()* below.

String Expressions

String expressions are made up of constant strings (characters surrounded by double quotation marks), variables (cell names, which refer to the cells's label strings or expressions), and string functions. Note that string expressions are only allowed when entering a cell's label string, not its numeric part. Also note that string expression results may be left or right flushed or centered, according to the type of the cell's string label.

\# Concatenate strings. For example, the string expression

A0 # "zy dog"

displays the string *the lazy dog* in the cell if the value of A0's string is *the la*.

Built-in String Functions

@substr(se,e1,e2)

Extract and return from string expression se the substring indexed by character number *e1* through character number *e2* (defaults to the size of *se* if beyond the end of it). If *e1* is less than 1 or greater than *e2*, the result is the null string. For example,

@substr ("Nice jacket", 4, 7)

returns the string *e jac*.

@fmt(se,e)

Convert a number to a string. The argument *se* must be a valid printf(3) format string. *e* is converted according to the standard rules. For example, the expression

*@fmt ("**%6.3f**", 10.5)*

yields the string ***10.500***. *e* is a double, so applicable formats are e, E, f, g, and G. Try *%g* as a starting point.

@sval(se,e)

Return the string value of a cell selected by name. String expression *se* must evaluate to a column name (*A-AE*) and *e* must evaluate to a row number (0-199). If *se* or *e* is out of bounds, or the cell has no string value, the result is the null string. You can use this for simple table lookups. Be sure the table doesn't move unexpectedly!

@upper(e)
@lower(e)

Will case the string expression to upper or lower.

@capital(e)

Will convert the first letter of words in a string into upper case and other letters to lower case (the latter if all letters of the string are upper case).

@ext(se,e)

Call an external function (program or script). The purpose is to allow arbitrary functions on values, e.g. table lookups and interpolations. String expression *se* is a command or command line to call with *popen(3)*. The value of *e* is converted to a string and appended to the command line as an argument. The result of *@ext()* is a string: the first line printed to standard output by the command. The command should permit exactly one output line. Additional output, or output to standard

error, messes up the screen.

ext() returns a null string and prints an appropriate warning if external functions are disabled, *se* is null, or the attempt to run the command fails.

External functions can be slow to run, and if enabled are called at each screen update, so they are disabled by default. You can enable them with ^*T* when you really want them called.

A simple example:

@ext ("echo", a1)

You can use *@ston()* to convert the *@ext()* result back to a number. For example:

@ston (@ext ("form.sc.ext", a9 + b9))

Note that you can built a command line (including more argument values) from a string expression with concatenation. You can also "hide" the second argument by ending the command line (first argument) with # (shell comment).

@coltoa(e)

Returns a string name for a column from the numeric argument. For example:

@coltoa(@mycol-1)
@nval(coltoa(@mycol-1), @myrow+1)

Built-in Financial Functions

Financial functions compute the mortgage (or loan) payment, future value, and the present value functions. Each accepts three arguments, an amount, a rate of interest (per period), and the number of periods. These functions are the same as those commonly found in other spreadsheets and financial calculators.

@pmt(e1,e2,e3)

@pmt(60000,.01,360) computes the monthly payments for a $60000 mortgage at 12% annual interest (.01 per month) for 30 years (360 months).

@fv(e1,e2,e3)

@fv(100,.005,36) computes the future value for of 36 monthly payments of $100 at 6% interest (.005 per month). It answers the question: "How much will I have in 36 months if I deposit $100 per month in a savings account paying 6% interest compounded monthly?"

@pv(e1,e2,e3)

@pv(1000,.015,36) computes the present value of an a ordinary annuity of 36 monthly payments of $1000 at 18% annual interest. It answers the question: "How much can I borrow at 18% for 30 years if I pay $1000 per month?"

Built-in Date and Time Functions

Time for *sc* follows the system standard: the number of seconds since 1970. All date and time functions except *@date()* return numbers, not strings.

@now Return the current time encoded as the number of seconds since the beginning of the epoch (December 31, 1969, midnight, GMT.)

@dts(e1,e2,e3)

@dts(9,14,1988) converts the date September 14, 1988 to the number of seconds from the epoch to the first second of 9/14/88, local time. For example, *@date(@dts(12,14,1976))* yields Tue Dec 14 00:00:00 1976

The month should be range from 1 to 12, the day should range from 1 to the number of days in the specified month, and the year should range from 1970 to 1999.

@tts(e1,e2,e3)

@tts(8,20,45) converts the time 8:40:45 to the number of seconds since midnight, the night before. The hour should range from 0 to 23; the minutes and seconds should range from 0 to 59.

The following functions take the time in seconds (e.g. from @now) as an argument and return the specified value. The functions all convert from GMT to local time.

@date(e) Convert the time in seconds to a date string 24 characters long in the following form:

Sun Sep 16 01:03:52 1973

Note that you can extract parts of this fixed-format string with *@substr()*.

@year(e) Return the year. Valid years begin with 1970. The last legal year is system dependent.

@month(e) Return the month, encoded as 1 (January) to 12 (December).

@day(e) Return the day of the month, encoded as 1 to 31.

@hour(e) Return the number of hours since midnight, encoded as 0 to 23.

@minute(e) Return the number of minutes since the last full hour, encoded as 0 to 59.

@second(e) Return the number of seconds since the last full minute, encoded as 0 to 59.

Spreadsheet Update

Re-evaluation of spreadsheet expressions is done by row or by column depending on the selected calculation order. Evaluation is repeated up to *iterations* times for each update if necessary, so forward references usually work as expected. See *set* above. If stability is not reached after ten iterations, a warning is printed. This is usually due to a long series of forward references, or to unstable cyclic references (for example, *set A0*'s expression to *A0+1*).

FILES

/usr/local/lib/sc/tutorial.sc tutorial spreadsheet

SEE ALSO

bc(1), dc(1), crypt(1), psc(1)

BUGS

Top-to-bottom, left-to-right evaluation of expressions is silly. A proper following of the dependency graph with (perhaps) recourse to relaxation should be implemented.

Only one previous value is saved from any call of *@ext()*. If it is used more than once in a spreadsheet and external functions are enabled and later disabled, the last returned value pops up in several places.

On some systems, if the cell cursor is in column 0 with *topline* enabled (so the current cell is highlighted), or if any cell in column 0 is highlighted, the corresponding row number gets displayed and then blanked during a screen refresh. This looks like a bug in curses.

Many commands give no indication (a message or beep) if they have null effect. Some should give confirmation of their action, but they don't.

AUTHORS

This is a much modified version of a public domain spread sheet originally authored by James Gosling, and subsequently modified and posted to USENET by Mark Weiser under the name *vc*. The program was subsequently renamed *sc*, and further modified by numerous contributors, Jeff Buhrt of Proslink, Inc. ({sequent, uunet}!sawmill!prslnk!buhrt) and Robert Bond of Sequent, prominent among them. Other contributors include: Tom Anderson, Glenn T. Barry, Gregory Bond, Stephen (Steve) M. Brooks, Peter Brower, John Campbell, Lawrence Cipriani, Jim Clausing, Dave Close, Chris Cole, Jonathan Crompron, David I. Dalva, Glen Ditchfield, Sam Drake, James P. Dugal, Paul Eggert, Andy Fyfe, Jack Goral, Piercarlo "Peter" Grandi, Henk Hesselink, Jeffrey C Honig, Kurt Horton, Jonathan I. Kamens, Peter King, Tom Kloos, Casey Leedom, Jay Lepreau, Dave Lewis, Rick Linck, Soren Lundsgaard, Tad Mannes, Rob McMahon, Chris Metcalf, Mark Nagel, Ulf Noren, Marius Olafsson, Gene H. Olson, Henk P. Penning, Rick Perry, Larry Philps, Eric Putz, Jim Richardson, Michael Richardson, R. P. C. Rodgers, Kim Sanders, Mike Schwartz, Alan Silverstein, Lowell Skoog, Herr Soeryantono, Tim Theisen, Tom Tkacik, Andy Valencia, Adri Verhoef, Rick Walker, Petri Wessman, and Tim Wilson.

Appendix B
PSC *man* page

PSC man page

NAME

psc - prepare sc files

SYNOPSIS

psc [-fLkrSPv] [-s *cell*] [-R *n*] [-C *n*] [-n *n*] [-d *c*]

DESCRIPTION

Psc is used to prepare data for input to the spread sheet calculator sc(1). It accepts normal ASCII data on standard input. Standard output is a *sc* file. With no options, psc starts the spread sheet in cell A0. Strings are right justified. All data on a line is entered on the same row; new input lines cause the output row number to increment by one. The default delimiters are tab and space. The column formats are set to one larger than the number of columns required to hold the largest value in the column.

Options

-f Omit column width calculations. This option is for preparing data to be merged with an existing spreadsheet. If the option is not specified, the column widths calculated for the data read by psc will override those already set in the existing spreadsheet.

-L Left justify strings.

-k Keep all delimiters. This option causes the output cell to change on each new delimiter encountered in the input stream. The default action is to condense multiple delimters to one, so that the cell only changes once per input data item.

-r Output the data by row first then column. For input consisting of a single column, this option will result in output of one row with multiple columns instead of a single column spread sheet.

-s cell Start the top left corner of the spread sheet in *cell*. For example, -s B33 will arrange the output data so that the spread sheet starts in column B, row 33.

-R n Increment by *n* on each new output row.

-C n Increment by *n* on each new output column.

-n n Output *n* rows before advancing to the next column. This option is used when the input is arranged in a single column and the spread sheet is to have multiple columns, each of which is to be length *n*.

-d c Use the single character *c* as the delimiter between input fields.

-P Plain numbers only. A field is a number only when there is no imbeded [-+eE].

-S All numbers are strings.

-v Print the version of psc

SEE ALSO

sc(1)

AUTHOR

Robert Bond

Appendix C
XgrabSC *man* page

XgrabSc *man* page

NAME

xgrabsc - grab rectangular screen images and store in files

SYNOPSIS

xgrabsc [-d display] [-gGknrvwz] [-i windowId] [-o outputFile]
[-s seconds] [-S seconds] [-b percent] [-A andBits] [-O orBits]
[-BHDFCPcelRWXZ2]

DESCRIPTION

xgrabsc lets you grab arbitrary rectangular images from an X server and writes them to standard output in a variety of formats.

Command line options also allow reduction of colormaps, halftoning and dithering of color images, and direct mapping of color images to monochrome.

The default output format is gray-scale non-encapsulated PostScript.

OPTIONS

-d displayName
Use an alternate display. If no display is specified on the command line, xgrabsc looks for the environment variable **DISPLAY** for the name of the display and screen to grab from. Note that you must have permission to access the display on another computer.

-g Silence the bell that is normally rung while the screen is being accessed.

-G Enable ringing of the bell

-i windowID
Dump the window with the given ID.

-k Select the window under the mouse when the Control key is pressed. This option is normally used in getting images of menus. Pop up the menu, optionally move the pointer to the window containing the menu, and strike the Control key to begin the dump.

-r Dump the entire screen (root window).

-z Use rubber-band rectangle to select region to grab. This is the default. You must have a mouse to use this option.

-w Use xwd style window selection and dump selected window. You must have a mouse to use this option.

-n Inhibit server grabs. Normally xgrabsc will "grab" the server so that the screen is frozen while a rectangle is selected and the image is extracted. If the screen is not frozen, rubber-banding may cause video droppings on portions of the screen that are changing.

-o output-file
Write output to *output-file* instead of standard output. The output-filename, minus directory and extension, is used as the internal name for the image in formats supporting image names. PostScript, xwd, pixmap and bitmap formats all support image names.

-s seconds
Sleep for *seconds* seconds before commencing operation. This should be used if you need some time to get the target image ready.

-S seconds
Sleep for seconds seconds after window/rectangle selection. This is commonly used to pop up menus after a window has been selected but before xgrabsc takes its snapshot.

-v Display processing information on standard error output (stderr).

-b percent
Brighten or darken the image by *percent*. Percentages are given as integers. As in xloadimage, 100 is the base and a larger number will brighten the image while a smaller number will darken the image.

-A andBits
Clear all colormap bits up to the given plane. This has the effect of darkening the image somewhat and shrinking the apparent depth of the image (and, consequently, the size of the color table). *AndBits* should be in the range [1-8] inclusive.

-O orBits
Set all colormap bits up to the given plane. This brightens the image somewhat and also shrinks the apparent depth of the image. When both *-A* and *-O* are specified, ANDing will occur before ORing.

-B Convert the source color image to a monochrome bitmap. All colors falling below the average color intensity are mapped to black. Others

are mapped to white.

-R Reverse the colors in the image. The bits of each color used in the image are inverted.

-H Convert the source color image to a halftoned monchrome bitmap. Resolution is maintained by increasing the size of the image by a factor of four on both axes.

-D Convert the source color image to a dithered monochrome bitmap. This is like halftoning, but resolution is sacrificed to keep the resulting image the same size as the original. The matrix dithering algorithm used with this option is most suitable for line-drawings and text. For more complex graphics the *-F* option is recommended.

-F Convert the source color image to a dithered monochrome bitmap with the *Floyd-Steinberg* algorithm.

-C Write output in PostScript format using the *colorimage* operator for color printers. Color to grayscale conversion is bundled into the output so you can actually use either color or cheaper grayscale printers. For monochrome displays, the *-P* option will give more compact output, and is the preferred format.

-P Write output in PostScript format for monochrome printers. The number of bits per PostScript sample is determined by the depth of the image.

-c Suppress PostScript image run-length encoding. PostScript output is normally compressed to minimize the size of output. If your printer can't handle compressed output, you should use this switch. You must also select the type of PostScript output you want to have with the *-P* or *-C* options.

-e Create Encapsulated PostScript output, rather than normal vanilla PostScript. This adds EPSF header comments and removes all scaling and translation of the image. You must also select the type of PostScript output you want to have with the *-P* or *-C* options.

-l Use landscape layout (11 x 8.5) for PostScript output. This has no effect on other forms of output, and it is ignored if Encapsulated PostScript output is requested.

-W Write output in xwd format.

-X Write the output in X Bitmap format if the image is black and white, or

X Pixmap format if the image is gray or color.

-Z Write output in a format suitable for loading into the puzzle program (see example below).

PROCESSING ORDER

It is helpful to know the order of processing when multiple processing options are given on the command line.

Processing is done in five phases: 1) set up, 2) obtain image, 3) process colors, 4) poly->monochrome conversions, and 5) output conversion.

The set-up phase includes processing command-line options, sleeping, connecting to X-Windows, freezing the screen, and grabbing the mouse if necessary.

If the mouse is grabbed for rubber-banding, an upper-left-corner cursor is displayed until the left mouse button is pressed. A lower-left corner cursor is then displayed while drawing rubber-rectangles until the mouse button is released.

If the mouse is grabbed for *xwd*-style window selection, an *xwd*-style cursor is displayed until the left mouse button is pressed.

The mouse is then released.

The bell is then run and the image is pulled from the screen.

Following the image-grab, the bell is run twice and the screen is released.

If the image is not monochrome, the color manipulation functions are then applied in this order: brighten, AND, and OR, reverse.

Only one polychrome to monochrome conversion is allowed. If none of these is chosen, the color table of a polychrome image is compressed in preparation for output conversion.

The output stream is then opened and the image is written in the selected output format.

ENVIRONMENT VARIABLES

XGRABSC specifies command line arguments to be processed before those actually entered on the command line.

DISPLAY specifies the name of the display that xgrabsc should grab from.

EXAMPLES

The simplest form of use, giving PostScript output, is

```
xgrabsc > outfile.ps
```

To write output in PostScript format and send to the printer, use

```
xgrabsc | lpr
```

It is sometimes helpful to brighten an image somewhat before it is formatted for PostScript output. E.g., to brighten by 30%

```
xgrabsc -Pb 130 | lpr
```

If your printer supports color, and your display is color, you can have xgrabsc generate color output instead of gray scale:

```
xgrabsc -C | lpr
```

The default PostScript output attempts to scale the image so that it will all fit on one page, and is centered on the page. If you are grabbing images to include in documents, such as with FrameMaker, you should ask for Encapsulated PostScript output with the *-e* switch. For example:

```
xgrabsc -eC -o image1.eps
```

Encapsulated PostScript files may be printed on a printer, but be aware that xgrabsc does not include the *colorimage* operator in Encapsulated PostScript files, so color dumps can only be printed on color printers. If you know something about PostScript programming, you can edit a non-EPS dump file and extract the colorimage operator from it and insert it into your EPS file and then send the modified file to the printer.

To select an entire window, write output in puzzle format and read into the puzzle program, use the commands

```
xgrabsc -wZ >outfile.pzl
puzzle -picture outfile.pzl
```

To have xgrabsc sleep for three seconds before rubber-banding, display processing information, and have the result displayed with xwud,

```
xgrabsc -Wvs3 | xwud
```

To grab an image from another server and then reduce the colormap to three bits by ANDing, use

```
xgrabsc -dother:0.0 -A5 -X >outfile.xpm
```

You will, of course, have to go to the other machine to select the image with that machine's mouse.

LIMITATIONS

Colormaps larger than 256 entries are not currently supported. This means that it won't work with your fancy 24-bit display. Use xwd and the xwd2ps utility for now.

The default screen visual is used as the visual for the image. Visuals are associated with particular windows, and xgrabsc pretends ignorance about any windows but the root.

This software has been tested with StaticGray and 8-plane PseudoColor on DECStations (using both UWS 2.2 and X11 Release 4). It has also been tested with 8-plane PseudoColor on Sun SparcStations and various other platforms using X11 Release 4.

X11 Pixmap format is rather verbose. You may want to run large images through the compress utility before storing them in a file. E.g.,

```
xgrabsc -X | compress >outfile.xpm.Z
```

AUTHOR

Bruce Schuchardt
Servio Corporation
bruce@slc.com

ACKNOWLEGEMENTS

Some of the source code for xgrabsc came from the xloadimage project by Jim Frost (jimf@saber.com) and others. Jim's copyright has been included both here and in the source code.

The idea for using run-length encoding for PostScript output came from the xwd2ps project by Robert Tatar and Craig A. McGowan, as did the colorimage hack for monochrome display devices.

COPYRIGHT

SEE ALSO

X(1X), xhost(1), xwd(1X), xwud(1X), xwd2ps(1X), xloadimage(1X), xpm(1X), xpr(1X), puzzle(1X), compress(1), uncompress(1)

Appendix D
Xgrab *man* page

Xgrab *man* page

NAME

xgrab - interactive front for xgrabsc, an X-Windows image grabber

SYNOPSIS

xgrab

DESCRIPTION

xgrab lets you grab arbitrary rectangular images from an X server and writes them to files or commands (such as lpr) in a variety of formats.

xgrab is a front for the xgrabsc program. Read the man page for xgrabsc for a description of the options presented by xgrab.

After selecting options from the various categories presented, press the OK button to have xgrab run xgrabsc to let you grab an image from the screen. After you press OK, xgrab's window will disappear and xgrabsc will gain control until the grabbing process is finished. Afterwards, the xgrab window will reappear.

OPTIONS

Xgrab responds to the standard application options, such as *-display*. See the man page for X for a complete list.

RESOURCES

Xgrab relies on the installation of the XGrab resource file. This file is named XGrab.ad in the xgrabsc source directory. Copy it to /usr/lib/X11/app-defaults/XGrab, or place it in your private application resource directory, prior to running xgrab.

The XGrab resource file contains a complete collection of all the widgets used in the xgrab window. Global resources, such as default font and color, are at the bottom of the file.

EXAMPLES

The *To Command* output option may be used to pipe xgrabsc output to programs.

The most common commands are lpr for Postscript output, and xwud for X-Window Dump output. Programs that do not accept piped input should not be used in *To Command*.

LIMITATIONS

See the limitations listed in the xgrabsc man page.

AUTHOR

Bruce Schuchardt
Servio Corporation
bruce@slc.com

COPYRIGHT

SEE ALSO

X(1X), xgrabsc(1X), xhost(1), xwd(1X), xwud(1X), xloadimage(1X), xpm(1X), xpr(1X)

Appendix E
MMag *man* page

Mmag *man* page

NAME

mmag - X Window System Screen Magnification Program (Motif version)

SYNOPSIS

mmag [-toolkit option ...]

DESCRIPTION

Mmag is an application which allows square regions of the screen to be magnified, and which shows the pixel value and RGB value for each magnified pixel.

MENU

Use the left mouse button to activate the *Actions* pull down menu.

Grab — grabs the pointer. The cursor will change to a crosshair. A square box the size of the region to be magnified will track the cursor. Press MB1 (the left mouse button on a right-handed mouse, or the right mouse button on a left-handed mouse) to magnify the desired region. Press MB3 (the right mouse button on a right-handed mouse or the left mouse button on a left-handed mouse) to ungrab the pointer.

Zoom In — halves the resolution.

Zoom Out — doubles the resolution.

Grid On/Off — toggles a grid overlay.

Clear — clears the magnifier window.

Quit — exits the application.

NOTES

Use *Zoom In* and *Zoom Out* to adjust the dimensions of the magnifaction box.

Resizing the window, affects the size, and not the number of magnified pixels.

Looking at pixel values:

When there is an image in the magnifier display area, holding the mouse button down (MB1) with the cursor in the display area will result in the pixel value (for the pixel at the current cursor location) being displayed at the bottom of the magnifier window, as well as the RGB value for that pixel.

BUGS

If you specify a size (with the *-geometry* flag) that is smaller than the normal/default size, there may not be enough room in the window for the pixel value and RGB value display strings.

AUTHOR

Danny Shapiro
Digital Equipment Corporation
Workstation Systems Engineering
Palo Alto, CA 94301

Philip Schneider
Digital Equipment Corporation
Advanced Technology Development
Palo Alto, CA 94301

Appendix B:

XFCV Data Analysis and Visualization Systems

Installation Guide

(The reverse of this page is blank.)

XFCV Data Analysis and Visualization System

Installation Guide

April, 1992

This manual describes the installation procedures of the XFCV Data Analysis and Visualization System on the specified hardware/software platform(s).

Revision/Update Information:	V2.1 (supersedes V2.0 documentation)
Platform:	Sun SPARCstation (SPARCstation-1 through SPARCstation-2)
Software Prerequisites:	SunOS 4.1.1 (Solaris 1.0) w/ OpenWindows V2.0/V3.0 and DECwindows Motif for the SUN SPARCstation V1.0 or greater

April 1992

The information in this document is subject to change without notice and should not be construed as a commitment by Clarkson University. Clarkson University assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with terms of such license.

The following are trademarks of Digital Equipment Corporation: DECwindows, Digital, ULTRIX, VAX, VMS, DEC

Motif, Open Software Foundation, OSF, and OSF/Motif are trademarks of the Open Software Foundation.

PostScript is a registered trademark of Adobe Systems, Inc.

Open Windows, SPARCstation, Sun, SunOS, Solaris, Sun View, and Sun Workstation are trademarks of Sun Microsystems, Inc.

OPENLOOK and UNIX are registered trademarks of UNIX System Laboratories, Inc.

Contents

Preface

The *XFCV Data Analysis and Visualization System Installation Guide* provides details pertaining to the installation of the **XFCV Data Analysis and Visualization System.**

Intended Audience

This manual is intended for persons who will be responsible for the installation and maintenance of the **XFCV Data Analysis and Visualization System.**

Document Structure

This manual is organized into two major parts. Chapter 1 provides the details of tasks which must be performed prior to installation. Chapter 2 covers the installation procedure and post-installation tasks.

Conventions

Convention	Meaning
Ctrl/x	A sequence such as Ctrl/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 x	A sequence such as PF1 x indicates that you must first press and release the key labeled PF1, then press and release another key or pointing device button.
MB1, MB2, MB3	Indicates that you press a mouse button (MB). The ordering of the buttons on the mouse depends on the orientation of the mouse: for a right-handed mouse, MB1=left-most button, MB3=right-most button; for a left-handed mouse, MB1=right-most button, MB3=left-most button.
RET	A key is shown enclosed to indicate that you press a key on the keyboard.
...	In examples, a horizontal ellipsis indicates one of the following possibilities: - Additional optional arguments in a statement have been omitted. - The preceeding item or items can be repeated one or more times. - Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed or there is not sufficient room to show all items.
()	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[]	In format descriptions, brackets indicate that whatever is enclosed is optional; you can select none, one, or all of the choices.
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.

boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason.
italic text	Italic text represents information that you can vary in system messages (for example, Internal error *number*). Italics can also specify references to other sources of information (e.g., "refer to the *XFCV Installation Guide* ..."). Italic text is also used to show user input, in contrast to system output, in examples showing a dialog with the system.
UPPERCASE TEXT	Uppercase letters indicate that you must enter a command (for example, enter RUN XFCV). Uppercase letters can also indicate the name of a command, the name of a file, etc.
numbers	Unless otherwise noted, all numbers in the text are assumed to be decimal. Nondecimal radixes - binary, octal, or hexadecimal - are explicitly indicated.

Chapter 1

Preparing for the Installation

1.1 System Prerequisites

XFCV requires the following hardware/software components:

Sun SPARCstation (SPARCstation-1 throught SPARCstation-2) or compatible SPARC clone

16M main memory (minimum - 24M is recommended)

8 plane color graphics adapter

Color display (16" minimum)

SunOS V4.1.1 or higher

Open Windows V2 or V3 (recommended) or MIT X11 Release 4 with all MIT patches.

DEC DECwindows Motif for the SUN SPARCstation V1.0 or higher (required only for recompiling the visualization components)

Tape drive (1/4" or 8mm, depending of distribution requested).

Color PostScript printer (for printing screen images - optional)

1.2 Disk space requirements

The *XFCV Data Analysis and Visualization System* requires the following disk space for installation:

System executables and runtime modules	6.4M
Example data sets	0.2M
System sources	2.2M
System documentation (printable)	5.3M

The bare minimum space required for the system is 6.4M. The other components are optional. There is no specific location in which XFCV should

be installed. For example, many systems install XFCV is /usr/local/xfcv or /home/local/xfcv.

Note that future release of XFCV are likely to require more disk space, so you should consider that when choosing a location for the installation.

1.3 Preparing an area for XFCV

At this point, you should login as *root* in order to prepare an area in which to install XFCV. You will then need to locate a paritition on the system with sufficient available disk capacity to install the XFCV components you have decided to install. To determine the amount of free disk space on the system, use the *df* command (the % is the c-shell prompt - only type the df command:

% df

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/sd0a	9679	5663	3049	65%	/
/dev/sd1a	158994	136471	6624	95%	/usr
/dev/sd0g	151399	75692	60568	56%	/home

Choose a partition which has sufficient space. In this case, /home is the only partition on this system with sufficient space to hold the XFCV system.

If you have a prior version of XFCV installed, you should either remove it or rename it to something other than xfcv. For example, if the previous version of XFCV was installed in /home/local/xfcv:

% mv /home/local/xfcv /home/local/xfcv.old

Now, create a new area for this version of XFCV:

% mkdir /home/local/xfcv

Finally, to proceed with the installation, define an environment variable to point to the area just created for XFCV and move to that area:

% setenv XFCVHOME /home/local/xfcv
% cd $XFCVHOME

You are now ready to install the XFCV system from tape.

Chapter 2

Installing XFCV

2.1 Loading XFCV media

Once you have prepared the system for the installation of XFCV, you should now load the tape device with the distribution media. Depending on the type of tape drive, you need to open the drive door and insert the tape (on 8mm drives, you will need to push the drive door button and wait until the door opens, which may be up to a few minutes). You should determine the name of the tape drive on your system at this point (it is typically something like /dev/nrst0 or /dev/nrst1).

Enter the following commands to load the installation procedure from the tape:

```
% mt -f /dev/nrst0 rewind
% tar xvf /dev/nrst0
```

This will rewind the tape and restore the installation procedure into the current directory.

2.2 Running the XFCV Installation Procedure

To continue the installation, run the installation procedure as follows:

```
% ./install_xfcv
```

Follow the dialog as shown below. Output from the installation procedure will be shown in normal case, while input from you will be shown in italics:

```
XFCV Installation Procedure V2.1

What is the name of the tape device you will be using (example: /dev/nrst0)?

/dev/nrst0

Where will you be installing the software (example: /home/local/xfcv)?

/home/local/xfcv
```

Which subsets do you wish to install:

1	System executables and runtime modules	6.4M
2	Example data sets	0.2M
3	System sources	2.2M
4	System documentation	5.3M
5	All of the above	
6	None of the above	

Subsets [5]? *5*

Preparing to install selected subsets. Is the tape drive online and ready? *y*

Rewinding tape...

Installing selected subsets in "/home/local/xfcv" ...

Installation complete. Rewinding tape...

Remove distribution from tape drive.

XFCV installation complete. Refer to Post-Installation tasks to finish installation.

At this point, the XFCV components you have selected have been installed on disk. There are a few tasks that must be done manually to complete the installation. These are covered in the next section. If you any error messages were displayed during the installation, you will need to return the the beginning of the installation and trace all the steps you performed to verify that they were correct and then repeat the installation. If you still obtain errors, call the support number listed on the invoice that was shipped with the system.

Remove the tape and return it to its case for protection.

2.3 Post-Installation Tasks

The are a few tasks which must be performed manually, following the installation of the software from tape. The following steps assume that your current working directory is still $XFCHOME.

XFCV, as it is built on top of OSF/Motif, requires key code symbol translation tables which are not typically resident on SunOS under OpenWindows. You need to determine whether there is an existing translation table already installed on your system and add the OSF/Motif one to it. If none already exists, then you must install the one distributed with XFCV.

First, check for the existence of the directory */usr/lib/X11*:

```
% ls /usr/lib/X11
```

If the directory does not exist, you will need to create it via:

```
% mkdir /usr/lib/X11
```

If it exists, check to see if there is an existing *XKeysymDB*:

```
% ls -l /usr/lib/X11/XKeysymDB
```

If so, then save the existing one, and make a new one by appending the XFCV XKeysymDB to it:

```
% mv /usr/lib/X11/XKeysymDB /usr/lib/X11/XKeysymDB.old
% cat X11/XKeysymDB /usr/lib/X11/XKeysymDB.old > /usr/lib/X11/XKeysymDB
```

If there is no existing *XKeysymDB*, then use the XFCV distributed file:

```
% cp X11/XKeysymDB /usr/lib/X11
```

The installation is now complete. Refer to the *XFCV Data Analysis and Visualization System User's Guide* for information on running the system.

(The reverse of this page is blank.)

Appendix C:

XFCV Data Analysis and Visualization System

Release Notes

(The reverse of this page is blank.)

XFCV Data Analysis and Visualization System

Release Notes

April, 1992

This manual lists the changes and new features to the XFCV system. It also discusses known problems and deficiences, along with workarounds, where available.

Revision/Update Information:	V2.1 (supersedes V2.0 documentation)
Platform:	Sun SPARCstation (SPARCstation-1 through SPARCstation-2)
Software Prerequisites:	SunOS 4.1.1 (Solaris 1.0) w/ OpenWindows V2.0/V3.0 and DECwindows Motif for the SUN SPARCstation V1.0 or greater

April 1992

Contents

Preface

The *XFCV Data Analysis and Visualization System Release Notes* provides details pertaining to changes, new features, and known problems with the current release of the **XFCV Data Analysis and Visualization System.**

Intended Audience

This manual is intended for all users of the **XFCV Data Analysis and Visualization System.**

Document Structure

This manual is organized into two major parts. Chapter 1 provides the details of changes and new features in this release. Chapter 2 provides information on the known problems and deficiences of the system, including workarounds where available.

Conventions

Convention	Meaning
Ctrl/x	A sequence such as Ctrl/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 x	A sequence such as PF1 x indicates that you must first press and release the key labeled PF1, then press and release another key or pointing device button.
MB1, MB2, MB3	Indicates that you press a mouse button (MB). The ordering of the buttons on the mouse depends on the orientation of the mouse: for a right-handed mouse, MB1=left-most button, MB3=right-most button; for a left-handed mouse, MB1=right-most button, MB3=left-most button.
RET	A key is shown enclosed to indicate that you press a key on the keyboard.
...	In examples, a horizontal ellipsis indicates one of the following possibilities: - Additional optional arguments in a statement have been omitted. - The preceeding item or items can be repeated one or more times. - Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed or there is not sufficient room to show all items.
()	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[]	In format descriptions, brackets indicate that whatever is enclosed is optional; you can select none, one, or all of the choices.
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.

boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason.
italic text	Italic text represents information that you can vary in system messages (for example, Internal error *number*). Italics can also specify references to other sources of information (e.g., "refer to the *XFCV Installation Guide* ..."). Italic text is also used to show user input, in contrast to system output, in examples showing a dialog with the system.
UPPERCASE TEXT	Uppercase letters indicate that you must enter a command (for example, enter RUN XFCV). Uppercase letters can also indicate the name of a command, the name of a file, etc.
numbers	Unless otherwise noted, all numbers in the text are assumed to be decimal. Nondecimal radixes - binary, octal, or hexadecimal - are explicitly indicated.

Chapter 1

Changes and New Features

1.1 Changes since V1.x

This section covers aspects of the XFCV system which have been changed since the Version 1.x releases.

- XFCV is now based on the OSF/Motif X Window toolkit. V1.x used the DECwindows XUI toolkit, which is being phased out by DEC in lieu of the OSF/Motif standard. In addition, the DECwindows XUI toolkit was only available on DEC hardware platform, which limited the portability of the system.

- New platform support:

  Sun SPARCstation, using OpenWindows V2 or V3 and the DECwindows OSF/Motif toolkit.

  MS-DOS/Intel 80x86 PC's using MS-Windows V3.0

- Spreadsheet function now uses public-domain *sc* spreadsheet. This replaces the older input/editing module which proved to be inadequate for larger data sets.

- Print Display function added using public-domain *xwd2ps*

- Added cluster shape display options (points & starburst)

- Enhanced on-line help

- Enhanced dynamic coloring algorithm

- Added vertical and horizontal scrollbars to main window to allow 'panning' the display. This is especially useful when zooming in on the display. Previously, there was no mechanism to view plots which were zoomed past the dimensions of the screen.

1.2 Changes since V2.0

The following changes have been made since V2.0 of XFCV:

- Cleaned up all dialog boxes, making them more consistent in appearance and behaviour.

- Changed Visualization Options dialog from using radio buttons to using option menus. The radio buttons required too much on-screen space, due to the ever increasing number of options. The new format will allow for more options to be added in the future, without requiring more dialog box space.

- Modified Edit Data sample dialog to show ID labels and added up/down arrows to more easily select sample #'s for editing.

- Modified default window fonts for xterm windows running analysis/scaling functions. This should make those dialog easier to read on systems with very small default fixed fonts.

- Modified the Transformations dialog to use an option menu rather than radio buttons for axis setting (see notes for Visualization dialog). Additionally made some cosmetic changes to other items in dialog, such as enclosing the orientation display in a box frame rather than having it appear to be 'floating' in the dialog box.

- Modified Print Display function to use public-domain *xgrab/xgrabsc* program to grab/print display. The older *xwd2ps* lacked various format and enhancement supported by *xgrabsc*. In addition, the user interface for *xgrab* is better. However, *xgrab/xgrabsc* do not support 24-bit displays, so for users running XFCV on 24-bit displays, *xwd2ps* will have to be used.

- Extended on-line help to include information on using on-line help, on the version of XFCV (including credits/copyright), tasks, and release notes.

- Modified display of all raw color numbers to show 0..255 rather than internal X color numbers (0..65535). Some users were confused by the use of two number schemes.

- Corrected numerous bugs carried over from V1.x.

1.3 New Features in V2.1

The following features are new to V2.1 of XFCV:

- Added exit dialog to allow the user an escape from exiting the system prematurely and warning/error dialogs to alert the user to errors.

- Added magnifier to system using public-domain, OSF/Motif based *xmag* (refered to as *mmag* to prevent confusion with MIT supplied *xmag*). This will allow the user to zoom in (on a pixel level) on areas of the display to better observe the coloring of points.

- Added Load/Save sample ID capability:

  This enables data set samples to be assigned class information and descriptive labels.

- Added *Customize Visualization Parameters* dialog to allow the customization of cluster coloring, adjustment of fuzzy tolerance, dynamic coloring, etc.

- Added file selection box and type/format dialog to *Spreadsheet* function to make operation more consistent with load/save functions and to support editing of sample ID data..

Chapter 2

Known Problems

There are some problems with XFCV V2.1 that appeared too late in the development cycle to be corrected. The following are a description of the known problems and various workarounds available.

2.1 Problems related to OpenWindows V2

Various problems exist in the Open Windows V2 X server and related software which affect the operation of XFCV.

Problem:

Popup dialog boxes do not have title bar or resize handles on the edges.

Explanation/workaround:

There is a problem in the way the Open Windows window manager interpretes window creation in that it does not place title bars/resize handles on windows it considers 'transient'. To some extent, this is a cosmetic issue. However, it does lead to some confusion for the user if a pop-up appears as a result of an errant selection. Without the title, it may be difficult to discern which dialog box one is looking at. Other problems deal with the resizing and relocation of the pop-ups. In most cases, dialog pop-ups will not need to be resized. To move a popup w/o a title bar, grab the pop-up near its edge and drag it to the desired location. We expect this problem to be resolved in a future version of XFCV (which will deal explicitly with window managers, like V2 of olwm, that do not implicitly put title/resize bars on pop-ups).

Problem:

When rotating the display, you may see a column of garbage appear.

Explanation/workaround:

This is a problem with the way the Open Windows V2 X server is handling the fonts. Either avoid rotations that cause the problem or switch to using the Open Windows V3 server or the MIT X11 Release 4 X server.

Problem:

Display sometimes does not redraw correctly after being occluded by another window.

Explanation/workaround:

The Open Windows V2 X server doesn't always deliver expose events to XFCV when it needs to redraw window contents, nor does it correctly save the window contents to allow the server to refresh the window correctly. To redraw the window, use the window manager's *Refresh* function to force an expose event. This will allow XFCV to redraw its window contents.

Problem:

Depending on the font used, you may notice random pixels left behind when moving sliders (such as the X, Y, Z rotation sliders).

Explanation/workaround:

This is a problem in the Open Window X server's handling of fonts. It is a cosmetic problem only. Use the window manager's *Refresh* function to force a redraw of the window and it will erase the errant pixels.

2.2 Problems related to Open Windows V3

Various problems exist in the Open Windows V3 X server and related software which affect the operation of XFCV.

Problem:

Popup dialog boxes do not have title bar or resize handles on the edges.

Explanation/workaround:

This is the same problem as was detailed for the Open Windows V2 X server. Refer to section 2.1 for more information.

Problem:

Display sometimes does not redraw correctly after being occluded by another window.

Explanation/workaround:

This is the same problem as was detailed for the Open Windows V2 X server. Refer to Section 2.1 for more information.

Problem:

Depending on the font used, you may notice random pixels left behind when moving sliders.

Explanation/workaround:

This is the same problem as was detailed for the Open Windows V2 X server. Refer to Section 2.1 for more information.

2.3 XFCV specific problems

There are various bugs/annomalies in the behaviour of XFCV which may not give the expected results. Some of these are known problems which are expected to be fixed in a future patch release of XFCV, while others are the intended behaviour for this release.

2.3.1 Main window

The following problems exist in the main window display:

Problem:

The scroll bars on the main window are overly sensitive, thus warping the image too far, too quickly. This makes it difficult to accurately pan the window.

Explanation/workaround:

Known problem. We expect to allow the scrollbar sensitivity to be user defined in a future release of XFCV. For now, pan the display as close as possible using the slider or by pressing MB1 in the trough of the scrollbar (this will move the plot by 1/4 the default height/width of the window), then click on the arrows at either end of the scrollbar to make fine adjustments.

Problem:

All pop-up dialogs occlude the main window.

Explanation/workaround:

This is an interaction between XFCV and the window manager. Some window managers 'auto-place' all child windows of a client on top of the client. Because of this, all pop-ups were defined without explict placement. We expect this behaviour to be corrected in a future release of XFCV. For now, grab and move the dialog boxes to another location on the screen.

2.3.2 File menu

The following problems exist in the *File* menu:

Problem:

Upon starting XFCV, most menu selections are not selectable (greyed-out).

Explanation/workaround:

This is the intended behaviour. Until a PC coordinate data file (*.pcdata) is loaded, the other functions have no meaning and thus are not valid choices. After loading a .pcdata file and then a .class file, all menu items will be selectable.

Problem:

A new file was created, but when re-entering the file selector sometime later, it doesn't appear in the list of files.

Explanation/workaround:

Known problem. The file selection boxes do not automatically re-execute the filter in order to update the contents of the box. Use the 'Filter' button in the box to update the contents. We expect this problem to be resolved in a future patch release of XFCV.

2.3.3 Edit menu

The following problems exist in the *Edit* menu:

Problem:

The *Edit Data Points...* selection is not selectable upon starting XFCV.

Explanation/workaround:

Until a PC coordinate data file (*.pcdata) is loaded, 'Edit Data Points' is greyed-out, since it needs data to function. This is the intended behaviour.

2.3.4 Visualization Options

The following problems exist in the *Visualization Options* function:

Problem:

When selecting *Fuzzy color* and then selecting to assign cluster/class by *Combination (Fuzzy)*, the color of the fuzzy samples does not change as expected.

Explanation/workaround:

This occurs the first time you select assignment as *Combination (Fuzzy)* due to the XFCV not pre-computing the fuzzy coloration. To correct it, select fuzzy cluster/class assignment first, and then fuzzy coloring. We expect this problem to be corrected in a future patch release of XFCV.

Problem:

When selecting to show the cluster shape as *Starburst*, the lines all emminate from what appears to be the origin of the window to each point.

Explanation/workaround:

The starburst shape visualization only works after class information has been loaded. Until class membership information is known to XFCV, cluster centers are undefined and thus the lines are drawn from the only known center, ie, (0,0). We expect to prevent this from happening in a future release of XFCV by greying-out the selection when class information does not exist.

Problem:

When selecting to display the cluster/class assignments 'As assigned', everything is shown as '0'.

Explanation/workaround:

To define cluster/class for each sample you must either use the *Edit Data Points...* dialog or load a sample ID file. Until ID's are known, the assigned cluster/class is not known and default to '0'.

2.3.5 Transformations

The following problems exist in the *Transformations* dialog box:

Problem:

When selecting to display either both, negative, or no axis, the axis display in the small view box does not change.

Explanation/workaround:

This is the intended behaviour. The axis setting only affects the axis displayed in the main window. The axis in the small view box is intended to be a reference for the rotations and thus is not affected by the setting of the axis display, nor by the zoom/shrink setting.

Problem:

During transformations, the diplayed orientation of the X axis changes, relative to changes in the Y and Z axis, but then further transformations about the X axis are still performed relative to the original X axis and not the modified orientation.

Explanation/workaround:

This is a deficiency in the way the transformation functions are being performed. For the transformations to be correct, you must do them in order X, Y, Z. Thus if after you have modified the X, Y, and Z rotations, and want to go back and modify the Y rotation, you must reset the Z rotation to 0 before modifying the Y rotation. This was not the intended behaviour. It is anticipated that this will be fixed in future patch release of XFCV.

Note that this problem will also affect the proper operation of the animation of the plot.

2.3.6 Animate Plot

The following problems exist in the *Animate Plot* selection:

Problem:

After selecting to Animate Plot, the main window has a small area which is not redrawn, near the bottom of the menu bar. In addition, none of the menu options work when the plot is being moved.

Explanation/workaround:

This is a two-fold problem. First, it is intended that, for this release, that the animation of the display be 'modal', that is, that it preempt other activity. This also means that XFCV does not update any occluded area of its window (other than the main display) until the animation is complete, hence the menu bar problem mentioned. It is expected in a future release of XFCV that the animation will be controlled by another dialog box, which will allow you to stop the animation before it is complete.

2.3.7 Magnify Area

The following problems exist in the *Magnify Area* function:

Problem:

After 'grabbing' a region to be displayed, the area shown in the magnifier window does not fill the window.

Explanation/workaround:

This appears to be a problem with the Open Windows X server. To force the magnifier to use the full area, try to resize the magnifier by grabbing a resize handle and expanding the window slightly in the vertical direction.

Problem:

After exiting XFCV, the magnifier window is left on the screen.

Explanation/workaround:

The magnifier, for this release of XFCV anyway, is an external program. To remove the window, select *Quit* from the magnifier's *Actions* menu.

2.3.8 Customize Animation Parameters

The following problems exist in the *Customize Animation Parameters* dialog:

Problem:

The dialog box is not wide enough to display the dialog box title in its entirety and the box cannot be resized.

Explanation/workaround:

This is a cosmetic problem. It is expect to be fixed in a future patch release of XFCV.

2.3.9 Customize Visualization Parameters

The following problems exist in the *Customize Visualization Parameters* dialog:

Problem:

Upon changing the *Fuzzy color tolerance*, and clicking MB1 on the *Apply* button, there is no change in the display.

Explanation/workaround:

It may require the *Apply* button be selected twice for the new value to be recognized. This is not the intended behaviour. We expect to fix this problem in a future patch release of XFCV.

Problem:

When changing the color of a particular cluster/class when viewing the samples by *Fuzzy Color*, the fuzzy samples appear to 'warp' to a new color.

Explanation/workaround:

This isn't really a problem as much as it is a function of the way the fuzzy coloring works. Since the fuzzy color is a function of the colors of both classes to which a sample belongs, when you change the color of one of them, it will have an additive affect on the fuzzy color.

Problem:

Sometimes the fuzzy color tolerance will be displayed as something like '0.050000' and the field is hard to edit.

Explanation/workaround:

There is a problem in the way the %f output field descriptor is working under SunOS cc. The field may be editing by locating the cursor somewhere near the beginning of the field and using the Delete key to remove the extra text.

Problem:

After changing a cluster's color and using the up/down arrows, the color reverts to its previous color.

Explanation/workaround:

This is the intended behaviour. To make the color change permanent, use the *Apply* button to update the cluster's color before using the up/down arrows.

2.3.10 On-line Help

You may notice one or more of the following problems with the on-line help system.

Problem:

When invoking the online help for *Tasks*, the Additional Topics selections are compressed into a small region instead of being displayed using the full width of the window.

Explanation/workaround:

This appears to be a problem with the Open Windows window manager/X server. Try either closing the Help window and reselecting it or saving your changed data and exiting XFCV and rerunning it. This usually corrects the problem. The nature of the problem is being investigated.

Problem:

When using the *Search by Keyword*, it never finds any matching keywords.

Explanation/workaround:

Currently, no keywords are defined in the on-line help. We expect this problem to be corrected in a future patch release of XFCV.

2.3.11 PC plotting component

The following problems exist in the PC plotting component of XFCV:

Problem:

Can't enter a filename/path longer than about 20 characters. This is too limiting for Unix.

Explanation/workaround:

Acknowledged. We expect this will be fixed in a future patch release of XFCV. For now, specify filenames less than 20 characters.

2.3.12 Data scaling component

The following problems exist in the data scaling component of XFCV:

Problem:

Can't enter a filename/path longer than about 20 characters. This is too limiting for Unix.

Explanation/workaround:

See notes on same problem with PC plot component.

2.3.13 FCV clustering component

The following problems exist in the FCV clustering component of XFCV:

Problem:

Can't enter a filename/path longer than about 20 characters. This is too limiting for Unix.

Explanation/workaround:

See notes on same problem with PC plot component.

(The reverse of this page is blank.)